NANYANG TECHNOLOGICAL UNIVERSITY

UNIFIED FEATURE MODEL FOR THE INTEGRATION OF CAD AND CAX

UNIFIED FEATURE MODEL FOR THE INTEGRATION OF CAD AND CAX

CHEN GANG

SCHOOL OF MECHANICAL & AEROSPACE ENGINEERING

2007

UNIFIED FEATURE MODEL FOR THE INTEGRATION OF CAD AND CAX

CHEN GANG

2007

Unified Feature Model for the Integration of CAD and CAx

Chen Gang

School of Mechanical & Aerospace Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of
Doctor of Philosophy

2007

UNIFIED FEATURE MODEL FOR THE INTEGRATION OF CAD AND CAX

Submitted

by

CHEN GANG

SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

Presented to the
Nanyang Technological University
In fulfillment of the
Requirements for the
Degree of Doctor of Philosophy in Engineering
Nanyang Technological University

2002/2006

Acknowledgements

I would like to express my wholehearted gratitude to my supervisors Dr. Ma Yongsheng and Dr. Georg Lothar Thimm for their guidance and patience over the past four years.

I would like to thank my family for their encouragement and support.

Table of Contents

Page

Abstract

In the context of concurrent and collaborative engineering, the validity and consistency of product information become important. However, it is difficult for the current computer-aided systems to check the information validity because the engineers' intent is not fully represented in the product model. It is also not easy to check and maintain the consistency among related product models because information associations are not fully established.

This research proposes and develops a feature-based product modeling scheme for the integration of computer-aided applications, especially CAD and CAPP integration. The scheme extends the traditional feature concept to a flexible and enriched data type, unified feature, which can be used to support the validity maintenance of product models. The novelty of this research is that the developed unified feature scheme is able to support data associations and propagation of modifications across product development life cycle stages. Therefore, it is a significant contribution to feature level interoperability in future virtual enterprises and collaborative engineering. In the proposed scheme, unified features provide an intermediate information layer to bridge the gap between engineering knowledge and product geometry. Unified features are also used to maintain geometric and non-geometric relations across product models.

Major technical aspects of the developed scheme include:

- The identification of common properties and methods of different application features, and the definition of a generic, unified feature type that represents these common characteristics.
- A rule-based expert system is embedded into a feature-based product modeling system to handle non-geometric data and relations.
- A unified, multi-dimensional cellular model accommodates different geometric modeling requirements uniformly.
- Relations across the CAD and CAPP applications are identified and managed by a dependency network with a justification-based truth maintenance system.
- A database structure for sharing data and storing the inter-stage associations is developed.

The feasibility of the proposed unified feature modeling scheme is demonstrated with a prototype system and three case studies which involve conceptual design, detailed design, and process planning.

List of Figures

List of Tables

List of Publications

1. G. Chen, Y.-S. Ma, G. Thimm, and S.-H. Tang. Unified feature modeling scheme for the integration of CAD and CAx, *Computer-Aided Design and Applications*, Vol. 1, No. 1-4, pp. 595-602, 2004.
2. G. Chen, Y.-S. Ma, G. Thimm, and S.-H. Tang. Unified feature based integration of design and process planning, *Proceedings of the 34th International MATADOR conference*, Manchester, United Kingdom, pp. 63-68, 2004.
3. G. Chen, Y.-S. Ma, G. Thimm, and S.-H. Tang. Knowledge-based reasoning in a unified feature modeling scheme, *Computer-Aided Design and Applications*, Vol. 2, No. 1-4, pp. 173-182, 2005.
4. G. Chen, Y.-S. Ma, X.G. Ming, G. Thimm, Stephen S.G. Lee, L.-P. Khoo, S.-H. Tang, and W.F. Lu. A unified feature modeling scheme for multi-applications in PLM, *Proceedings of The 12th ISPE International Conference on Concurrent Engineering (CE2005): Research and Applications – Next Generation Concurrent Engineering: Smart and Concurrent Integration of Product Data, Services, and Control Strategies*, M. Sobolewski and P. Ghodous (Eds.), ISPE, Dallas, USA, pp. 343-348, 2005.
5. G. Chen, Y.-S. Ma, G. Thimm, and S.-H. Tang. Associations in a unified modeling scheme, *Journal of Computing and Information Science in Engineering*, Vol. 6, No. 2, pp. 114-126, 2006.
6. G. Chen, Y.-S. Ma, G. Thimm, and S.-H. Tang. Using cellular topology in a unified feature modeling scheme, *Computer-Aided Design and Applications*, Vol. 3, No. 1-4, pp. 89-98, 2006.

Chapter 1
Introduction

1.1 Background

Product development comprises several life cycle stages, such as conceptual design, detailed design, process planning, machining, assembly, etc. Computer-aided tools (called “CAx systems” hereafter) are commonly used to support activities associated to these stages. Usually, two engineering domains are separately managed, i.e. product and process domains. Recently, due to the drive of industrial globalization and mass customization, concurrent and collaborative engineering approaches gain importance. This trend leads naturally to the necessity of an integration of product and process domains as well as the integration of CAx systems. This research accommodates for this by a proposition of a new product modeling approach based on feature technology with the consideration of knowledge-based engineering, system integration, and collaboration.

A comprehensive mechanical product model consists of linked geometric and non-geometric data throughout all life cycle stages. A product model has to be constructed or analyzed iteratively using engineering knowledge from different aspects of expertise to fulfill requirements, such as functional or manufacturing requirements. In addition, life cycle stages are inter-related and mutually constraining. Any modification in one stage may provoke a chain of subsequent modifications to entities of the same or other stages. This propagation of changes requires the management of inherent relations within and among these stages.

1.2 Problem Statement

Traditionally, stand-alone CAx systems for individual stages produce separate models, such as a product design or a process plan. The existing CAx technology has difficulties in maintaining the integrity of the comprehensive product model as inter-stage data sharing is insufficiently supported, especially for non-geometric data. Furthermore, validity checking of product models is difficult as the engineering knowledge applied in product designs or process plans is usually not stored with the model as existing technology does not allow for this.

1.3 Research Goal

The goal of this research is to establish a paradigm in product modeling across multiple life cycle stages. The knowledge engineering and geometric modeling are integrated in a systematic and scalable manner. The paradigm must allow that multiple applications can share a consistent product model with supporting mechanisms to maintain its integrity and validity.

1.4 Research Scope

This research covers the following aspects:

- A data structure that is sufficiently comprehensive and flexible to be used by conceptual design, detailed design, and process planning applications. Properties and functional methods of the common data structure must allow dealing with associations between engineering knowledge and product geometry such that engineering modification propagation is supported with model validity checking.
- To represent engineering knowledge in a reusable and scalable manner in the product model. Knowledge driven product modeling with user-defined rules, exploration of alternative solutions, validity checking of modified product models, and embedding of engineering intent should be supported.
- To develop generic methods for maintaining the consistency of product models, including both geometric and non-geometric data.
- To develop algorithms for intra- and inter-stage propagation of modifications.
- To develop a prototype system and test with case studies to illustrate that the proposed modeling scheme and methods are feasible and expandable.

Note that the following two issues are not of concern here:

- The detailed processes or algorithms related to specific applications, such as algorithms for functional design or process planning;
- Feature recognition and conversion algorithms.

1.5 Contributions of this Research

The main contributions of this research are:

- A data structure – called unified feature – was defined to support data associations and sharing across conceptual design, detailed design and process planning stages.

- Engineering intent is explicitly stored in product models as associations among engineering knowledge, application-specific non-geometric entities, product structures and specifications.
- Inherent relations within a single and among applications are identified and stored.
- Algorithms for propagating modifications.
- Three case studies – involving conceptual design, detailed design, and process planning – are carried out with the developed prototype system.

Chapter 2
Literature Review

2.1 Introduction

Today, the requirements on product quality, time-to-market and manufacturing cost become more and more stringent. This leads to the wide use of concurrent and collaborative engineering approaches. These approaches require the integration of product data, process data, and the management of engineering knowledge in an effective and efficient manner. Hence, keeping the validity and consistency of product and process information is consequently emphasized.

This review starts with the current state of the art of concurrent and collaborative engineering (Section 2.2). In addition, three key technological areas are explored, i.e. application integration, enterprise resources planning, and knowledge-based engineering. Existing problems for concurrent and collaborative engineering are then identified.

Since the introduction of feature technology two decades ago, it was proven to be a useful tool to model engineering semantics as well as to maintain associations among geometric entities. Therefore, feature technology is instrumental to concurrent and collaborative engineering. In Section 2.3, feature-based product modeling is reviewed because the candidate believes that existing feature technology can be enhanced by a unified approach. Several fundamental issues of feature technology, which include feature definitions, geometric representation schemes, feature relations, validity checking, and feature-based application integration, are discussed.

In Section 2.4, issues with current feature technologies are identified and analyzed. Considering the research scope, challenges in the research arena are evaluated and the focus of this research justified.

2.2 The State of the Art Research on Concurrent and Collaborative Engineering

To introduce the background and context of the presented research, the current status of research on concurrent and collaborative engineering is reviewed in this section. The existing problems are identified and discussed.

2.2.1 Concurrent Engineering

As a basic methodology for product development and manufacturing, concurrent engineering was studied by academia and implemented in industry to compress time to market and cost. The basic concept is to achieve concurrency and close-loop in product development and manufacturing processes via tight application integration and parallel engineering. Concurrent engineering considers all life cycle issues simultaneously (Prasad [1]). The traditional sequential mode of product development has been changed into an iterative and evolutionary mode. On one hand, to realize the overall required product functions, many aspects need to be considered at the same time, such as the spatial, stability, and esthetic concerns in building design (Rosenman & Gero [2]). On the other hand, to fulfill product life cycle requirements, downstream stages, such as machining and assembly for mechanical products, need to be considered to optimize the product design (Xue & Yang [3], Xue et al. [4], Roucoules et al. [5], Feng & Song [6], Feng & Song [7]). These publications reveal the importance of the inherent relations across product life cycle stages.

2.2.2 Collaborative Engineering

Collaborative engineering supports distributed, multi-discipline, and multi-organization teams during the product development processes. This approach is motivated by the globalization of economy and boosted by the development of the Internet (Fuh & Li [8], Yang & Xue [9], Wang et al. [10]). This approach influences the development of product modeling systems. Consistency control among distributed but related applications is one of the main aspects of the influence.

In general, the consistency among distinct applications can be controlled through:

- Communications via agents (Rosenman & Wang [11]);
- CORBA-based distributed programming (Chan et al. [12], Pahng et al [13]);
- Common databases (Kung et al. [14]).

In particular, among designers working on the same component, the geometric consistency can be controlled by specifically developed naming mechanisms (Li et al. [15], Bidarra et al. [16], Lee et al. [17]). For designers working on different components of an assembly, assembly features can be used to control the geometric compatibility between mating components (Shyamsundar & Gadh [18]). These requirements need to be considered when developing a product modeling scheme.

2.2.3 Application Integration

Traditional application integration approaches focus on the geometric data sharing. For example, integrations between design systems and reverse engineering, rapid prototyping, coordinate measuring, mesh generation, virtual reality, process planning, and assembly systems have been widely studied (Varady et al. [19], Benko et al. [20], Starly et al. [21], Kramer et al. [22], Rezayat [23], Ma et al. [24], Fuh et al. [25], Noort et al. [26]). In these publications, the most common approach to support application integration is using a set of neutral formats, such as the Initial Graphics Exchange Specification (IGES) or the STandard for the Exchange of Product model data (STEP) (ISO 10303-42 [27]), for geometric data exchange. To support fully application integration, more comprehensive data sharing is needed than that provided by the existing IGES or STEP standards.

2.2.4 Enterprise Resource Planning

Developed from the initial inventory control, then subsequent material requirements planning (MRP) and manufacturing resources planning (MRP II), enterprise resource planning (ERP) is used to integrate all the separate enterprise information systems, such as design, manufacturing, material planning, finance, etc. (Giachetti [28], Umble et al. [29], Jacobs & Bendoly [30]). Many aspects of the ERP approach have been studied, such as the process integration in an ERP system, relations with the electronic commerce, supply chain, and product data management systems (Akkermans et al. [31], Park & Kusiak [32], Wang et al. [33], Ou-Yang & Chang [34]). The importance of ERP is widely recognized. However, the engineering data integration with ERP systems is a hurdle for its effective implementation. Although ERP is not in the focus of this research, preliminary results show the potential to integrate feature-based product models with process models via ERP.

2.2.5 Knowledge-Based Engineering

The product development process can be regarded as a set of decision making processes. Knowledge-based engineering (KBE) approaches are used in many CAx systems to support decision making, such as functional design, geometric constraint solving, parametric design, assembly oriented design, feature recognition, determining tooling components or layout, and process planning (Tor et al. [35], Lee & Kim [36],

Myung & Han [37], Zhang & Xue [38], Zha et al. [39], Zha et al. [40], Henderson [41], Lee et al. [42], Mok et al. [43], Sormaz & Khoshnevis [44], Park [45]). KBE approaches provide a better way than pure geometric modeling to embed engineering intent during product development processes.

2.2.6 Application Problems

Many problems persist in the implementation of concurrent and collaborative engineering, namely:

- The representation and processing methods for engineering intent. Many decisions made in the product development and manufacturing processes are supported by engineering principles, concepts, and rules. Product models are the reflection of and hence need to be verified based on such engineering intent. However, engineering intent is mainly represented as 'know-how' by individual engineers, or is only implicitly embedded in product data relations. The lack of intent representation has affected the product validation processes.
- Data consistency. Concurrent and collaborative engineering deal with separate but related applications. Theoretically, all CAx applications should operate on a common set of data so that the product engineering and management can be efficient to changes required. However, these applications have difficulties in sharing consistent and comprehensive product models.

2.2.7 Prospective Solutions

Many partial solutions to these problems were developed. For example, Unigraphics software streamlines commands to propagate changes of the shared model among collaborators (Unigraphics [46]). Complete data sharing is supported if collaborators use the same software. To support real time solid model sharing and modifications among diverse CAx systems, 'OneSpace Collaboration' software uses IGES or STEP as the neutral format (CoCreate [47]). Only geometric data sharing is realized. As analyzed in (Ma [48]), sharing of non-geometric data, such as parameters, constraints, and features among different CAx systems, is still an unresolved problem due to the lack of a commonly accepted standard (Pratt et al. [49]). As features are widely used in current CAx systems, and feature technology has the potential to overcome these problems, it is

selected as the foundation of this research. Major aspects of feature technology are reviewed next.

2.3 Feature-Based Product Modeling

Research on how features are defined, how features are used in single applications, and how features are used to integrate applications are reviewed in this section.

2.3.1 Historical Definitions of Features

Feature concept is flexible and can be used in many aspects of mechanical engineering (Shah & Mantyla [50]). Representatives of application independent feature definitions include:

- A region of interest in a part model (Wilson & Pratt [51])
- Any geometric form or entity that is used in reasoning in one or more design or manufacturing activities (Cunningham & Dixon [52])
- Generic shapes associated to certain properties or attributes and knowledge useful in reasoning about the product (Shah [53])
- Regions of an object that are meaningful for a specific activity or application (Vandenbrande & Requicha [54])
- A set of form elements with a functional meaning in a given application context that allows an association between shape and functionality (Martino et al. [55])
- A representation of shape aspects of a product that are mappable to a generic shape and are functionally significant for some product life-cycle phase (Bidarra & Bronsvoort [56])

These definitions reveal that features have two fundamental characteristics:

- Features relate to product geometry in a level higher than geometric and topological entities.
- They represent engineering intent.

2.3.2 The Fundamentals of Feature Technology

As building blocks of product models, three fundamental issues, which are geometric representation, data relation management, and model validation, must be addressed by feature technology. The relevant research is reviewed in this sub-section.

Geometric Representation Schemes

The mainstream geometric representation schemes used in feature-based modeling systems are boundary representation (B-rep) and constructive solid geometry (CSG) [50]. B-rep stores the part boundary, such as faces, edges, and vertexes, explicitly. Low level information is hence available but the data structure is complex for specification and modification. In contrast, CSG uses primitive solids and set operations to construct part geometry. Low level information is not available but the data structure is compact and easy for specification (Hoffman [57], Requicha [58]). Usually, these two schemes are combined, such as in (Venkataraman et al. [59], Roy & Liu [60], Gossard et al. [61]). Traditional B-rep and CSG usually represent only two-manifold solids. It is not easy to represent overlapping and multi-dimensional geometries in these schemes. This limitation makes them unsuitable to be used as geometric engine to support application integration. The cellular topology, due to its unique data structure, is more suitable to serve the integration purpose. Research on cellular model is reviewed later.

Relations in a Feature-Based Application Model

Features combine geometric and non-geometric entities. Therefore, compared with geometric models, more complex relations exist in feature models. Managing these relations, especially the non-geometric ones, is essential for the validity of a product model (Otto [62]). Relations in a feature-based product model can be classified as follows (Table 2.1):

- *Geometric relations*: Many publications focus on geometric relations in a feature model (Brunetti et al. [63], Shah & Rogers [64]). All these relations are explicitly declared and represented as geometric constraints, which maintain features' geometric integrity. However, unintentional feature interactions may also affect features' validity (Hounsell & Case [65], Bidarra et al. [66], Karinthi & Nau [67]). These interactions usually cannot be prevented by geometric or algebraic constraints. The geometric feature interactions can only be managed through the associations between the feature model and the geometric model.
- *Non-geometric relations*: Non-geometric relations refer to dependency relations involving non-geometric entities. For example, in process planning, a feature may remove the clamping faces or accessing faces required to machine another feature if a wrong machining sequence is used (Faheem et al. [68]). Furthermore, two features,

which do not spatially overlap, even belong to different product life cycle stages, may interact with each other (Regli & Pratt [69]). In addition, non-geometric relations also exist between features and non-geometric entities. For example:

(i) In design stage, functional-form matrixes, bipartite function-feature graphs, design flow chain and key characteristics, and mapping hierarchy are used to link features to product functions (Mukherjee & Liu [70], Feng et al. [71], Whitney et al. [72], Brunetti & Golob [73], Brunetti & Grimm [74]).

(ii) In process planning stage, features are related to non-geometric entities, such as machines, cutting tools, and machining processes (Sormaz & Khoshnevis [75], Khoshnevis et al. [76]).

The methods of representing non-geometric relations and using them to validate product models haven't been fully developed.

Table 2.1 Summary of research on relations in feature-based applications

Relation nature	Related entities	Representation	Source
Geometric relations	Between geometric entities	Geometric constraints	[63] [64] [116]
	Between features	Interaction constraints	[65] [66] [67] [81]
Non-geometric relations	Between features and the corresponding geometric entities	Features refer to the corresponding geometric entities	[54] [56] [79] [80]
	Between features	Not mentioned	[68] [69] [83]
	Between features and other non-geometric entities, such as functions, behaviors, assembly methods, machines, cutting tools.	Tables, graph, rules, etc.	[45] [70] [71] [72] [73] [76] [84] [89] [134]

Using Features to Maintain a Product Model's Validity

A product model must have a sound mechanism to check its validity. Compared to the strict validity maintenance mechanisms of B-rep or CSG, current feature-based modeling schemes are weak in this aspect. Rossignac suggested that feature's validity should be defined in terms of the referenced geometric entities, and of their existence, shape, and relations to other geometric elements of the model (Rossignac [77]). However, as analyzed before, features' validity must be checked in a broader scope. A feature model is valid if:

- The geometric and algebraic constraints specified on features are satisfied.
- The feature model is consistent with the geometric model (Geelink et al. [78], Laakko & Mantyla [79]). This consistency can also be used to validate engineering

intent (Mandorli et al. [80], Vieira [81], Martino et al. [82]) However, in these publications, engineering intent must be transformed into geometric, algebraic, or preliminary semantic constraints, such as the boundary or interaction constraints [56]. During the transformation processes, engineering intent may be lost.

- The non-geometric constraints specified among features are satisfied. For example, different machining sequences may influence the presence, shape, volume, and validity of machining features (Sharma & Hayes [83], [68], [69]). However, the representation, checking, and maintenance methods of inter-feature non-geometric constraints are immature.
- The features are consistent with the related non-geometric entities. For example, the presence of features or the values of feature parameters may be determined by functional requirements or machining conditions (Anderl & Mendgen [84], Oral & Cakir [85]).
- The feature model is consistent with the engineering intent of using these features.

The first aspect is the traditional constraint satisfaction problem, which has already been solved to a wide extent. Some researchers considered the second aspect but hardly touched on the other three aspects although they are equally important for product model validity.

2.3.3 Traditional Feature-Based Applications

This section reviews definitions, semantics, and usage of features in specific applications. Table 2.2 gives a summary of the semantics and geometric representations of application features.

Table 2.2 Summary of application features

Application	Semantics	Geometry	Current limitations	Source
Conceptual design	Representing the interacting parts. The interactions result in the state transitions of the parts, which are part behaviors that are used to realize product functions	Solid, surface, wireframe	Geometric representations need to be further studied; Representation of functions and behaviors in features are incomplete	[70] [86] [87] [88] [89]
Detailed design	Portions of a single part that materialize the conceptual design or are added for DFX purposes	Solid	Engineering intent of the used form features are not well specified	[92] [95] [96] [97] [98] [99]

Table 2.2 Summary of application features (continued)

Application	Semantics	Geometry	Current limitations	Source
CAE analysis	Portions of a component, which are significant in a particular type of analysis	Solid, surface, wireframe	Not included in this research	[90] [100]
Assembly design	Portions of mating parts that achieve the product functions or are created for modularization or enhancing the assemblability	Solid, surface	Functional relations across features are not well specified	[64] [72] [101] [102] [103] [104] [105] [106] [107] [145]
Process planning	Geometry created or used in manufacturing processes, which depends on manufacturing constraints or process planning preferences	Solid, surface	Machining features are not well related to their engineering intent	[54] [108] [109] [110] [111] [112] [114] [115] [131] [133] [134]
Assembly planning	Portions of mating parts that affect the assembly processes, sequence, stability, etc.	Solid, surface	Not included in this research	[116] [117] [118]

Conceptual Design Features (also Called Functional Features)

The conceptual design stage concentrates on determining basic principles and critical specifications of a product. The engineering intent in this stage is on the product's functions, structure, material, and other properties. Due to the lack of product details, at present, only a few researchers consider using features in this stage:

- Some functional feature definitions are non-geometric in nature. They are only used to explain the purpose of individual design objects (Brown [86], McGinnis & Ullman [87]).
- Other functional feature definitions connect product geometry with functions. For example, functional features were defined as the reasoning behind form features' usage (Wood and Ullmann [88]), or form features that are connected to product functions [70]. These two definitions do not represent interactions between components, which are crucial for function realization.
- Schulte defined functional features as surfaces, which interact with each other based on physical effects to fulfill product functions (Schulte et al. [89]). The specific arrangement and relative motions of faces are the main content of functional features (Figure 2.1). This definition preliminarily links the realization of product functions to the corresponding geometry and topology.

None of the above definitions fully fulfill the following two requirements:

- Functional requirements and function realization have to be represented by features.

- Incomplete, inexact, but critical product geometries have to be represented by features.

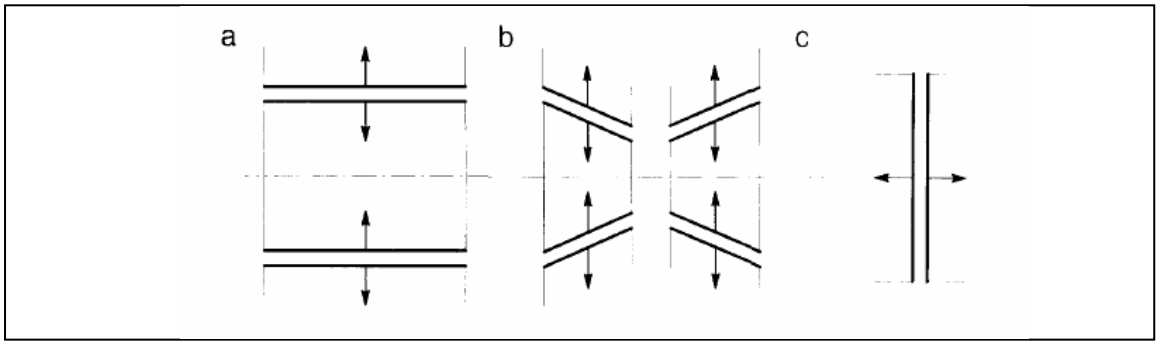


Figure 2.1 Three types of functional feature "frictional connection", from [89]
(Design function: input or output of torque/speed; Physical effect: dry friction; Geometry: a pair of separate faces with suitable material vectors)

Detailed Design Features

In the detailed design stage, the main task is specifying the detailed product geometry, topology, and other properties for production. The design intents include:

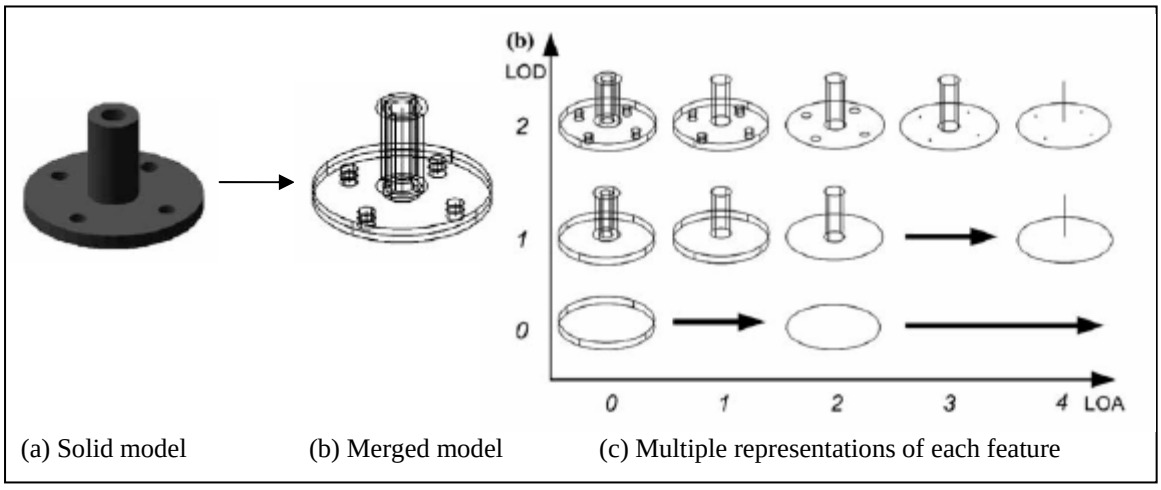
- Keeping consistent with the guidelines specified in the conceptual design, such as required functions or design patterns (Lee [90], Ma & Tong [91]). Stefano et al. used particular geometric characteristics in detailed designs to represent product functionality (Stefano et al. [92]). However, how to connect these geometric characteristics to the conceptual design for design validation is not mentioned.
- DFX considerations (Boothroyd et al. [93], Fazio et al. [94]). Most research on feature-based manufacturability or assemblibility analysis can be put into this category, such as the machining feasibility, complexity analysis, interference checking, or moldibility evaluation (Ong & Chew [95], Li [96], Chen et al. [97], Lockett & Guenov [98]).

However, usually only form features are used in the detailed design stage for product geometry construction. The purpose of these features is usually not clearly specified. A typical definition of form feature is "a shape macro that is constructed for convenience, with little connection to function or manufacturing" (Han & Requicha [99]). Form features hide the tedious geometric modeling and editing processes from designers. Most current feature technologies, such as feature relations and validity control, deal with form

features only. Since the purposes of traditional form features are not well specified, it is difficult to use them directly in application specific reasoning processes.

CAE Features

In CAE analysis stage, the engineering intent is to select portions of a detailed design that are significant in a specific type of analysis. For example, the suppressibility feature attributes can be specified for idealization purpose (Deng et al. [100]). Lee used idealization features to integrate design and CAE analysis [90]. These idealization features contain the necessary information for the construction of models in different detail or abstraction levels (Figure 2.2). The idealization features keep the consistency among these representations. Only the geometric and topological specifications of the idealization features are mentioned.



(a) Solid model (b) Merged model (c) Multiple representations of each feature

Figure 2.2 Each idealization feature contains multiple representations, from [90]

Assembly Design Features

The main design intent in this stage is achieving the required product functions through specifying relations between mating parts. Assembly features were traditionally used to represent geometric relations between mating parts (Chan & Tan [101], Case & Harun [102], Anantha et al. [103], [64]). The design intent of using these features and the corresponding geometric relations is not explicitly specified. In addition, these definitions are only suitable when form features in mating parts have already been created (Figure 2.3), i.e. in a bottom-up design approach.

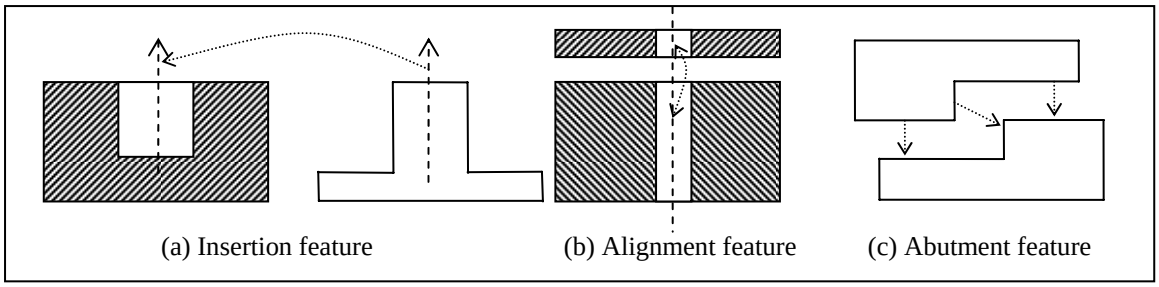


Figure 2.3 Assembly features, from [64]

However, for the top-down design processes, relations between parts may need to be specified when the detailed part geometry have not been completely designed (Ma et al. [104]). These relations can be defined as assembly features and used to assign responsibilities in collaborative design (Csabai et al. [105], Shyamsundar & Gadh [106]). The possibility of using assembly features to represent design intent is mentioned in (Deneux [107]). However, no details are given. Whitney suggested transforming design intent into the associations among key characteristics, the corresponding datum flow chain, and assembly features [72]. This approach is more appropriate for representing design intent than purely using geometric or algebraic constraints.

Machining Features

The origin of the feature technology probably lies in the search for a high level geometric representation to support process planning and CNC machining. Initially, only geometric aspects of machining features were considered. It was assumed that the purpose of machining features is implicitly embedded in the product model. For example, machining features are traditionally defined as volumes of material removed in machining operations (ISO 10303-224 [108]). Formalisms, such as attributed adjacency graph (Joshi & Chang [109]) or cell decomposition (Tseng & Joshi [110]), were proposed to represent machining features. Gradually, it was found that pure geometric methods may result in features, which are geometrically valid but invalid in the view of machining. Some researchers proposed using additional methods, such as the validity test in [54], to prune infeasible machining features. Features mentioned in this paragraph are only machining feature candidates given to the process planning application for further selection.

To generate feasible and optimal machining feature interpretations of a design, accessibility test and manufacturability analysis are used (Gupta & Nau [111], Chu & Gadh [112]) (Figure 2.4). In this way, although features are still geometrically defined, however, engineering intent is used to control the generation of feature instances (Marefat & Britanik [113]).

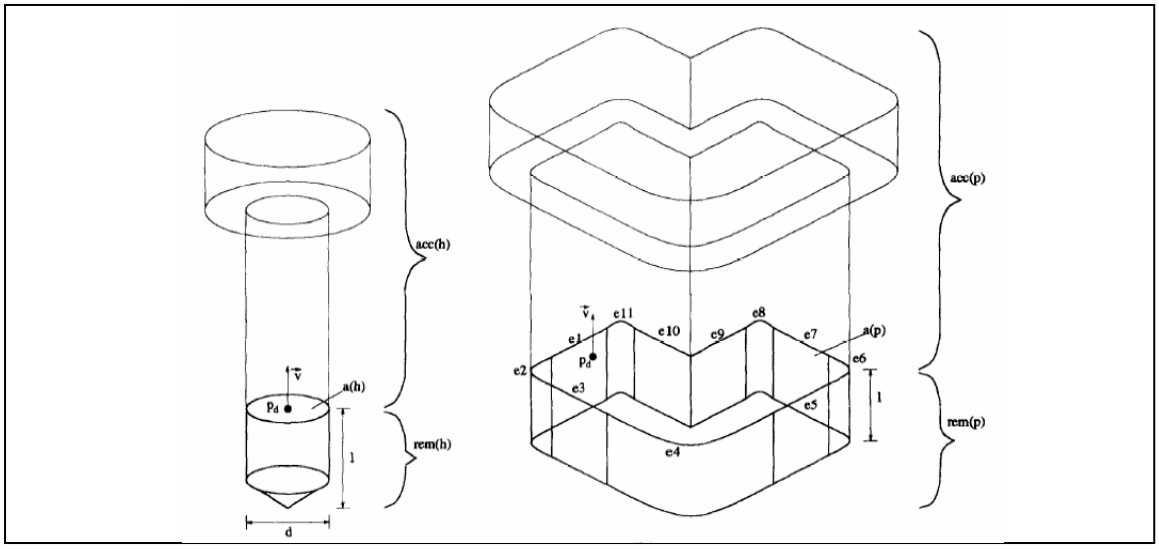


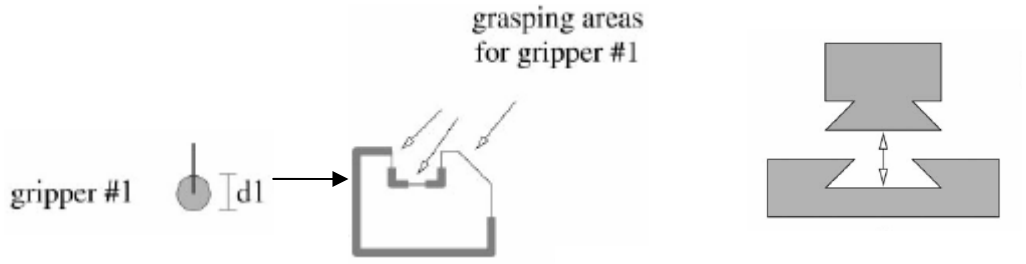
Figure 2.4 A drilling feature and an end-milling feature, from [111]

Features are also used in other process planning activities, such as fixturability analysis (Chu [112], Kumar et al. [114]), as well as in other manufacturing processes, such as casting (Stefano [115]).

Assembly Planning Features

Holland and Bronsvoort (Holland & Bronsvoort [116]) used handling features to represent feeding methods, feeding direction, fixturing methods, etc. Connection features are used to represent the insertion position, insertion path, tolerances, contact area, etc. The features (Figure 2.5) are generated by assembly planning activities, such as fixture planning, feeding planning, stability analysis, etc. Kim et al. studied the assemblies created by welding and riveting (Kim et al. [117], Kim et al. [118]). They used the associations among form features, geometric constraints, and joining methods to represent engineering intent. Non-geometric information is included in features and used to check whether the design specifications, such as DOF, are satisfied by the selected

joining methods. These two feature definitions link engineering knowledge, non-geometric entities, features, and product geometry explicitly.



(a) A handling feature corresponding to gripper #1 (b) A dove-tail connection feature

Figure 2.5 Handling and connection features, from [116]

Commonalities of Application Features

Reviews in this sub-section illustrate that application features are commonly regarded as constrained associations among a group of geometric and non-geometric entities. Application features must be tightly related to their purposes. Pure geometric definitions are insufficient. In detail, the commonalities of application features include:

- Features use parameters and attributes to describe geometric and non-geometric feature properties.
- Features refer to a specific set of geometric entities.
- Algebraic, geometric constraints as well as the relations between features and geometric entities have to be maintained in order for the geometric integrity and validity of a feature model to persist.
- Features correspond to particular engineering processes and intent.

These commonalities provide a basis for the generic definition of distinct application features.

2.3.4 Feature-Based Application Integration

Product development stages are inter-related and mutually constraining. The corresponding application models represent different aspects of the same product. When an application model is changed, the changes must be propagated to other related applications for checking and updating ([91], Dohmen [119], Park & Khoshnevis [120]). Therefore, application models must be connected or integrated, which can be achieved

by using feature technology. Many approaches have been proposed as shown in Table 2.3. These approaches are discussed as follows.

Table 2.3 Summary of research on feature-based application connection or integration

Approach		Connection mechanism	Connected or shared entities	Current limitations	Source
Feature conversion		Direct conversion between feature models	Feature volumes	Feasibility needs to be proved	[52] [122] [124] [125]
Feature recognition		Usually none, except for the JTMS used in [135]	Geometric entities	No connection after recognition	[41] [109] [110] [129] [130] [135]
Common feature definitions	Design by feature	Application features are used for product design	Application features	Limitations on designers	[50]
	Neutral features	All applications support neutral features	Neutral features	Lack of non-geometric associations	[108] [142]
Multiple-view feature modeling	Pre-defined central model	NMT model or feature model tree	Geometric entities	Lack of non-geometric associations	[90] [137]
	Incremental updating the central model	Cellular model, intermediate model or master model	Geometric entities; Or non-geometric entities that an application shares	Lack of non-geometric associations	[2] [55] [123] [138] [139] [140] [141] [175]

Feature Conversion

Some researchers suggested using feature conversion to directly convert one feature model into another feature model (Gao et al. [121], Bronsvoort & Jasen [122]). This approach is supposed to be able to maintain the consistency between feature models through direct connections. It can also use non-geometric information stored in the existing feature model to derive new feature models (Bronsvoort & Noort [123]). For example, Anderson and Chang used geometric reasoning to re-group design features into machining features (Anderson & Chang [124]). However, they assumed that part designs are created using only subtractive features, which can be regarded as primitive machining features. To convert positive feature models to negative feature models, Gurumoorthy et al. used a clipping and classification method (Subramani & Gurumoorthy [125], Subramani et al. [126]). However, they focus only on geometric relations between feature models.

Research on feature conversion makes useful explorations on directly linking feature models. However, the most common but also the most difficult type of feature

conversion exists between features that consist of different variations of the same set of geometric entities (Shah [127]). The many-to-many relations between feature elements make direct feature conversion almost infeasible. The non-geometric information carried by the feature model should be used together with the geometric information carried by the geometric model to derive a new feature model. How to propagate changes under this hybrid approach has not been fully studied yet.

Feature Recognition

The feature recognition approach recognizes features from solid models. Many researchers focus on this issue (Pal et al. [128], Kim [129], Regli [130], [110], [109], [41]). The multiple interpretations and feature interactions are main hurdles that hamper the use of this approach. Some researchers proposed using engineering intent to solve these two problems (Raman & Marefat [131], Li et al. [132], Gaines & Hayes [133], Stage et al. [134], [111], [54]). In other words, instead of pure geometric reasoning, features should be recognized in particular engineering contexts.

In addition, traditional feature recognition methods usually do not establish relations between the solid model and the recognized feature model. Any change in the solid model invalidates the whole recognized feature model. To solve this problem, Han proposed using volumetric interference checking and a Justification-based Truth Maintenance System (JTMS) to manage geometric relations between design features and machining features (Han [135]). Whenever design features are modified, the machining feature model is updated accordingly. However, non-geometric relations between design and machining features should be recorded and maintained too. For example, without any geometric modifications, just changing the tolerance specification of a design feature may invalidate the corresponding machining features (Zhou et al. [136]).

Common Feature Definition Approach

Due to the difficulties of feature conversion and recognition approaches, using the same set of features for multiple applications has been proposed as an alternative solution of application integration. This approach tries to use the same feature definitions for different applications. It intends to save the efforts of recognizing features and can also be used to support consistency control. The traditional design-by-feature approach can be regarded as belonging to this category. The initial purpose of design-by-feature

approach is considering the downstream requirements in the design stage as well as for application integration. The maintenance of the consistency between design and application models is simplified. However, due to the unreasonable limitations put onto the designers, this approach is rejected by industry. Another approach is developing a set of neutral features, which are supported by all applications. The representative is STEP [108]. However, current STEP standards focus too much on geometric representations of machining features.

Feature conversion, feature recognition, and design-by-feature approaches create sequential relations between feature models. However, in concurrent engineering, multiple views need to be maintained simultaneously to keep them consistent. To meet this need, the multiple-view feature modeling approach has been proposed.

Multiple-View Feature Modeling

Under multiple-view feature modeling approach, each feature model, which corresponds to a particular development stage, is a view of the whole product model. There are two main methods to realize multiple-view feature modeling.

- *Pre-defined central model*: Jha and Gurumoorthy proposed using a feature tree, which includes all the feature explanations of a product model, to achieve the application integration (Jha & Gurumoorthy [137]). Lee used a non-manifold topology to integrate design and analysis models [90]. Multiple levels of abstraction of each solid primitive are predefined in the corresponding feature classes and stored in a multi-dimensional model when features are generated. First, these two methods are geometric in nature, engineering intent is not represented. Second, the extensibility and applicability of these methods need to be proved since all feature interpretations or abstractions need to be predefined.
- *Incrementally updating a central model*: Bronsvoort et al. developed multiple-view feature models that encompass several product development stages. These feature models are integrated on the basis of a common cellular model (Bidarra et al. [138]). The cellular model is a non-manifold geometric model, which represents the combined product geometry from all stages (Dohmen et al. [139]). The relations among these views are established on the linking faces of features (Kraker et al. [140]). Suh and Wozny proposed using a set of fundamental features, which are

actually faces, edges, and vertexes, as a common layer to support multiple applications (Suh & Wozny [141]). Application features are generated by analyzing and re-grouping these fundamental features. Martino et al. suggested using an intermediate model to integrate applications [55]. The intermediate model consists of a set of common faces shared by features from different applications. Common faces are used to propagate modifications. These approaches consider only geometric relations between feature models.

In summary, it can be concluded that due to its unique nature, feature concept is capable of linking knowledge to geometry as well as integrating applications. However, current feature technology is insufficient for these two purposes. In the next section, these insufficiencies are identified and listed as issues that will be addressed in the presented research.

2.4 Research Issues and Challenges

To fulfill the requirements of concurrent and collaborative engineering approaches, the shortcomings of the current feature technology must be addressed. These shortcomings are listed as research issues. The challenges to solve them are analyzed in this section.

2.4.1 Research Issues

The following research issues are significant for the further development of feature technology.

Feature Interoperability

Diverse feature definitions make transferring or sharing of feature data among applications difficult. Traditional neutral feature definitions, such as [108], focus only on the representations of explicit feature geometries. Neutral representations of parameters and constraints are still immature. In addition, the engineering intent of using particular features cannot be completely represented by parameters and numerical constraints. Recent development of STEP tries to extend the portion of features which can be neutrally represented (Pratt & Srinivasan [142]). However, whether including all information into a single feature definition is a good idea is still uncertain [123].

Data associations can be used to represent feature's intent. An associative feature concept has been proposed to represented relations between different forms of geometric entities depending on specific applications (Ma & Tong [143], [91], Ma et al. [144]). Associative features also model the evolvement of features at different stages of product development. The modifications made in one stage may affect the validity of the model in another stage. This associative feature concept can be extended to include non-geometric associations. Furthermore, Ma et al. used assembly design features to represent essential relations between geometric entities, which may be assemblies, components, features, or faces ([104], Ma et al. [145]). These relations are design patterns generated in the conceptual design stage (Figure 2.6). Assembly design features can be regrouped and derived from specific viewpoints, such as functional design, assembly planning, or manufacturability analysis.

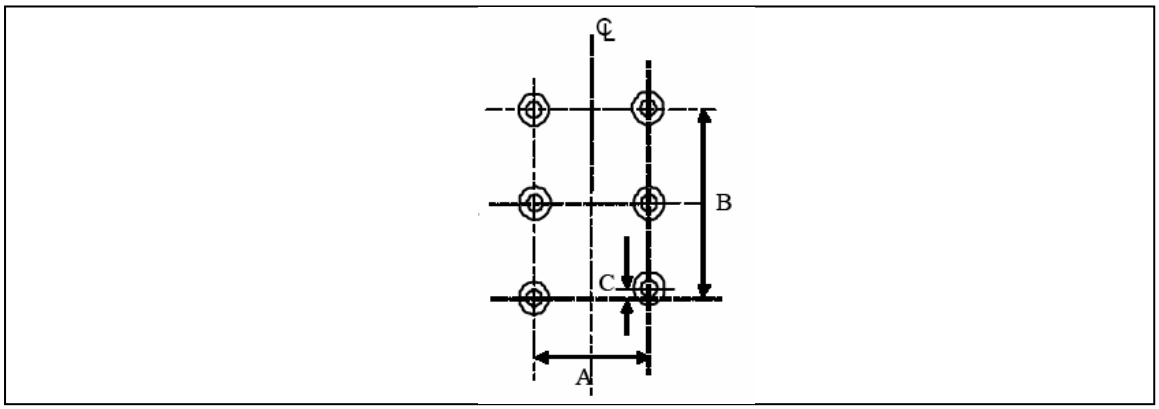


Figure 2.6 An assembly design feature: guide pin pattern, from [104]

In process planning domain, the resource adaptive feature concept has also been proposed to represent associations between machining volumes and the used cutting tools, machines, and setups ([131], [133], [134]). Thes resource adaptive feature definitions are not purely geometric any more. The machining specific entities, e.g. machines or cutting tools, are explicitly defined in feature classes as attributes or constraints. These associations are generated by reasoning processes, such as the machining sequence determination, setup determination, tool selection, etc. These associations participate in the representation of process planners' intent.

More research efforts are needed to develop a neutral and complete representation of features' intent. Another issue is that the traditional neutral feature definitions are usually rigid. The flexibility and scalability are hence limited.

Design Intent Representation and Management

Numerical constraints (geometric or algebraic) are traditionally used to represent design intent (Shah et al. [146]), but they are mostly restricted to detailed geometry. Numerical constraints cannot represent design intent flexibly and completely due to the following reasons:

- Numerical constraints represent only a subset of possible solutions in the view of realizing design intent. Other solutions may be represented by different parameters and constraints.
- Certain relations cannot be represented by numerical constraints. For instance, two design features may be related because they are used together to realize a product function. These non-geometric relations justify the features' presence or determine the values of feature's properties.

How to represent design intent in a complete and explicit manner is an unresolved issue.

Major decisions are usually made in the conceptual design stage. Integrating the conceptual design into the whole product model enables the comprehensive and explicit design intent representation (Gui & Mantyla [147], Henderson [148], Chandrasekaran et al. [149]). These publications indicated that:

- The downstream life cycle stages will benefit from the functional description of the product.
- The redesign and design modification processes will benefit because alternative solutions are stored in the product model explicitly.

Many methods have been proposed to represent the relations between product functions and physical structures. Kusiak et al. used a rule-based system to decompose customer requirements and functions, to map requirements to functions, and to record alternative solutions (Kusiak et al. [150]). Some researchers identified the necessity of using part behavior to bridge the gap between abstract functions and physical objects (Umeda et al. [151], Qian & Gero [152], Welch & Dixon [153]). They indicated that:

- Many functions of mechanical products are generated through the interactions between parts.
- Part behavior describes the state transitions of a part and is driven by part interactions.
- Functions and behaviors represent deeper design knowledge than geometry, topology, parameters, and constraints.

In addition, the vagueness of the product geometry in the conceptual design stage has also been addressed by using non-manifold geometric modeler and configuration spaces (Guan et al. [154], Wong & Sriram [155]). However, this is still an unresolved issue. Comparing the review in this section to those in Section 2.3.3 about definitions of conceptual design features, it is clear that there is a lot of work need to be done before the feature technology can be used to really support conceptual design as well as to link conceptual design to other stages for design intent representation.

Besides product functions, DFX is another kind of design intent. Some design specifications are generated to ease activities in the downstream stages. The intent behind these specifications should be represented too. For example, Thimm et al. proposed a graph-based method for automatic machining sequence generation and tolerance analysis for rotational parts (Thimm et al. [156], Thimm et al. [157]). A design dimension tree is used to generate ideal datum hierarchy trees and to measure real process plan efficiency. The structure of the datum hierarchy tree underlying a process plan is used to generate heuristics for more efficient design dimensioning schemes. Several heuristics are generalized based on the analysis of the relations between the design dimension specifications and the machining sequences. Examples of the heuristics are minimizing the datum changes and reflecting the manufacturing precedence constraints in the design dimensioning scheme. These heuristics serve the purpose of specifying better design dimension scheme for process planning and should also be recorded in the product model.

Multiple-View Geometric Representation

Feature geometries may overlap. In addition, their representations may obey different geometric modeling requirements, such as using solid, surface, or wireframe representation. In a multiple-view, feature-based product modeling scheme, incorporating these multi-faceted and diverse representations into a single geometric

model and keeping them consistent are unresolved issues. The multi-dimensional, non-manifold cellular topology provides a solution. Following the pioneer work of Weiler (Weiler [158], Weiler [159]), in which the radial edge structure was proposed to represent non-manifold geometries, how to use the non-manifold geometric model to store canonical forms of the original objects even they are not on the final boundary were discussed in (Masuda [160], Crocker & Reinke [161]). Non-manifold geometric models can be used to integrate applications ([90], Sriram et al. [162]). However, these publications did not fully apply the multi-dimensional, non-manifold topology to feature-based product modeling processes. Bidarra et al. proposed using a cellular model to support multiple-view feature-based product modeling process (Bidarra et al. [163], [138]). However, their research scope is confined to 3D features only. The cellular topology is also used for collaborative design (Lee et al. [164], Wu & Sarma [165]) and efficient feature recognition (Woo [166]).

It can be seen that the multi-dimensional cellular topology has the capability to realize the geometric integration of feature-based applications. However, this has not been fully realized yet. In addition, how to propagate geometric modifications, and in turn, how to maintain geometric consistency among different dimensional feature models are still research issues need to be addressed.

Persistent Representation for Non-Geometric Associations

In a single stage, the review in Section 2.3.2 shows that non-geometric relations exist and are important. However, usually, they are not well formulated and maintained for the reason that they usually cannot be fully represented as numerical constraints. Furthermore, it is the engineering intent that actually controls the generation of a product model ([45], [134]). The associations between engineering intent and the corresponding features have to be established and managed for product validation. Dependency networks provide a general solution. Kusiak and Wang used dependency relations to represent constraints on design variables (Kusiak & Wang [167]). Four types of constraints are mentioned: equation, qualitative constraints, computer-based procedures, and influence rules. Park and Cutkosky used precedence, constraint, and abstraction links to model dependency relations during the collaborative engineering processes (Park & Cutkosky [168]). Eastman analyzed how to use the dependency network to determine the influence scope of a design modification (Eastman [169]).

Non-geometric relations exist across application models as well. For example, associations between functional requirements in the conceptual design and geometric constraints or tolerance specifications in the detailed design must be maintained. Gorti and Sriram discussed the mapping of the functional relations among components to the corresponding spatial relations (Gorti & Sriram [170]). In such a way, alternative spatial relations for the same functional relations can be found. Ranta et al. suggested using generic ontologies to connect product development stages (Ranta et al. [171]). Common function based associations among geometric entities and the possible mapping of abstract functional requirements to geometric constraints are discussed. Roy et al. explained that part specifications usually depend on product functional requirements (Roy & Bharadwaj [172], Roy et al. [173]). Spatial relations cannot represent product functions completely. Energy transfer relations between part faces are also essential in describing product functions and determining product specifications, such as fit types, tolerances, and surface finishes. Bronsvoort and Noort identified some non-geometric relations among feature models in a multiple view modeling system [123]. However, they use only geometric relations to connect the detailed design with the assembly and manufacturing planning views. In addition, the connection mechanism between non-geometric data is not clear. In these publications, inter-stage non-geometric relations are usually established via direct links among specific entities. A more systematic and scalable method for non-geometric data sharing need to be developed (Figure 2.7).

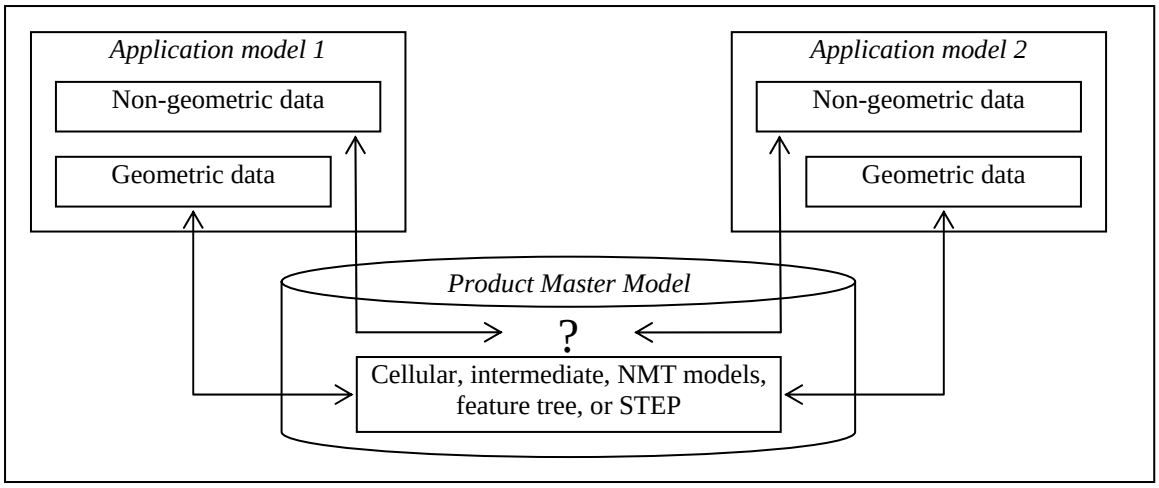


Figure 2.7 The lack of non-geometric associations between application models in the current multiple-view feature-based modeling schemes

Hoffman and Arinyo proposed a generic architecture to establish and maintain associations between application models (Hoffman & Arinyo [174], Hoffman & Arinyo [175]). The basic idea is that each client view deposits a part of their data into the master model and associates their private data to the public data. When a modification happens, the master model notifies the associated clients, who are responsible for maintaining the consistency. However, no implementation details are given.

2.4.2 Challenges to Solve the Existing Problems

Based on the highlighted issues, the research challenges are analyzed as follows.

How to Unify Different Application Features

Generally, features have two portions: geometric representation and non-geometric representation. The geometric representation can be unified based on the well accepted B-rep or cellular schemes. The engineering intent representation is difficult to be unified because they are application specific. Till now, the feature unification is unsuccessful. A generic, complete, and flexible feature definition, which can represent the commonalities of different application features, is needed.

How to Integrate Knowledge-Based Methods with CAx Tools

The full integrations of the KBE and the CAx systems (especially CAD systems) has not been realized yet ([91], Roller & Kreuz [176], Penoyer et al. [177]). The following problems need to be solved:

- Representing engineering intent using KBE;
- Using engineering knowledge to drive product modeling or process planning;
- Associating engineering knowledge with product designs or process plans;
- Most of the KBE systems used in CAx systems are one-way, i.e. from knowledge to decisions, such as product configurations or machining methods, etc. How to use engineering knowledge to verify product designs or process plans should be studied.

Establishing and Maintaining Non-Geometric Relations

The first challenge is to identify the major intra- and inter non-geometric relations in feature-based, multiple-view product models. The second challenge is to find suitable methods to represent and manage these non-geometric relations for the purpose of

maintaining the validity and consistency of product models. With the addition of these non-geometric relations, how to effectively propagate modifications and determine their influence scopes under the multiple-view product modeling approach are also challenges.

2.5 Summary

Feature technology is widely accepted as an indispensable part of CAx systems. Feature technology has the potential to bridge engineering semantics and geometry and eventually integrate multiple applications into a collaborative, associated, and knowledge driven enterprise information system. However, from the viewpoint of maintaining the validity and consistency of product models, current feature technology still has severe shortcomings:

- Different feature definitions and their data structures make transferring and sharing data between applications difficult. A generic and flexible data structure is necessary.
- Engineering intent is not well represented and maintained.
  (i) The engineering intent proper to each application needs to be identified.
  (ii) A method to represent and maintain engineering intent in product models is needed.
- Keeping the geometry of feature models consistent is difficult due to the different characteristics of feature geometry in distinct CAx applications.
- Non-geometric relations within and among application models are not properly represented and exploited.
- The lack of methods that determine the influence scope of modifications, especially across development stages, makes change propagation inefficient.

After a thorough literature review, a feature-based unification approach is regarded as suitable to bridge engineering semantics and product geometry and eventually integrate multiple applications.

2.6 Proposed Research Approach

To address the above mentioned shortcomings, in this research, a unified systematic CAx integration approach supporting multiple applications across product life cycle stages is proposed via feature-based interoperability scheme and mechanisms connecting different information granularity levels, i.e. from geometry, features to engineering

knowledge, based on a novel unified feature concept. Information association and unification are focused as the basic strategies in this research. Association is necessary because product development is such a complicated process. Many aspects need to be considered with many factors involved. Without complete and explicit data associations, engineering changes in CAx systems require tedious, error-prone, and usually manual processes, even with parametric and constraint-based design approaches. In contrast, unification is necessary because in reality, diverse CAx systems coexist. They have different data structures. Without a generic but also flexible product modeling scheme, comprehensive interoperability among CAx systems and further effective data consistency validation are difficult. From the proposed approach and strategies, research tasks are then identified as follows.

2.6.1 Extending Feature Definitions to Support Engineering Intent Embedment and Data Sharing among Applications

Traditional features are specifically defined for certain computer applications, and used mainly to represent parameterized geometric shapes. This is insufficient. First, a generic, flexible, and scalable feature definition is needed such that a common object class data structure can be used for different applications. Second, besides geometry, non-geometric feature data needs to be generically defined. The new feature definition should be used as an intermediate information layer to associate product geometry and engineering knowledge consistently and explicitly.

In this direction, some research works reported are related to associative features, resource adaptive features, etc. ([104], [91], [131], [118]). Extended feature types and ontology schemes are also reported in ([142], [74]).

2.6.2 Engineering Intent Representation and Management in Product Models

Engineering intent representation has been a hot research area for some time already but a concrete foundation has not been established yet. This research is intended to solve this problem systematically with emphasis on design intent representation and management in the conceptual design stage. Research in ([123], [73], [171]) preliminarily explores the methods to link conceptual designs to detailed designs. For the integration mechanisms between knowledge engineering methods and computer aided product design tools, some conceptual frameworks are proposed in ([176], [177]).

2.6.3 Inter-Stage Non-Geometric Relations

Inter-stage relations can be classified as geometric and non-geometric ones. The method to deal with inter-stage co-plane relations has been suggested by [138].

This research pays more attention to non-geometric relations among stages. They need to be identified and managed because they are crucial for consistency control among product models. Feature-level associations can be used to represent these non-geometric relations.

2.6.4 Multi-Dimensional Geometric Modeling

To accommodate different types of features that could be non-manifold and in different dimensional degrees, multi-dimensional geometric models must be integrated. As discussed in Section 2.4.1, the multi-dimensional cellular model can accommodate different geometric representation and modeling requirements uniformly. It can represent intermediate or overlapping geometries. It can also be used to keep the original forms of features regardless whether they are on the final boundary of the part or not. The feasibility of this approach has been studied in ([90], [163]).

2.6.5 Representation of Dependency Relations

Complete dependency relations must be maintained for effective and efficient management of engineering changes. Dependency network based on JTMS can be used to record and manage the dependency relations in product modeling processes. Choosing a Truth Maintenance System (TMS) approach is because it can accommodate different types of dependency relations uniformly, such as numerical constraints, antecedent to consequent relations in engineering rules, and feature objects dependency relations in feature recognition or conversion processes.

Chapter 3

Unified Feature-Based Product Modeling Scheme

3.1 Introduction

To support data sharing and associations across product development stages, this chapter introduces a solution framework that entails major class definitions, database structure, as well as integration and reasoning mechanisms based on a unified feature concept. The proposed scheme has the following main aspects:

- At each stage, a feature-based product model is used to represent product information. Engineering knowledge behind product designs and process plans is represented explicitly in the product model. To support the integration of applications and the incorporation of engineering knowledge, a feature class definition that can generically represent the common properties as well as the required methods throughout product life cycle stages is studied first; that is the proposed unified feature concept. By defining this common feature class, all other specific application features can be derived according to object oriented polymorphism approach. In such a way, product data that is shared among conceptual design, detailed design, and process planning applications follows a common class model.
- A mechanism to associate data across stages (represented by the corresponding applications) is developed based on the JTMS method (Forbus & Kleer [178]) to create a coherent product information model.
- An algorithm to propagate modifications across applications to maintain the validity of all product models.

To start with, the definition of unified feature is presented in this chapter due to its pivot role in the whole scheme. The integration of engineering knowledge and detailed geometry, i.e. knowledge embedment with cellular model is introduced in Chapter 4. Application feature definitions based on the unified feature concept are implemented for design and process planning in Chapter 5. The association mechanism and the change propagation algorithm are introduced in Chapter 6.

3.2 Unified Feature Concept

A unified feature is defined as a set of constrained associations among a group of geometric and non-geometric entities. As identified in Section 2.3.3, the commonalities of application features, such as conceptual design features, detailed design features, and process planning features, are defined in the unified feature class as generic fields and methods. Figure 3.1 gives the generic definition using a UML diagram (Booch et al. [179]).

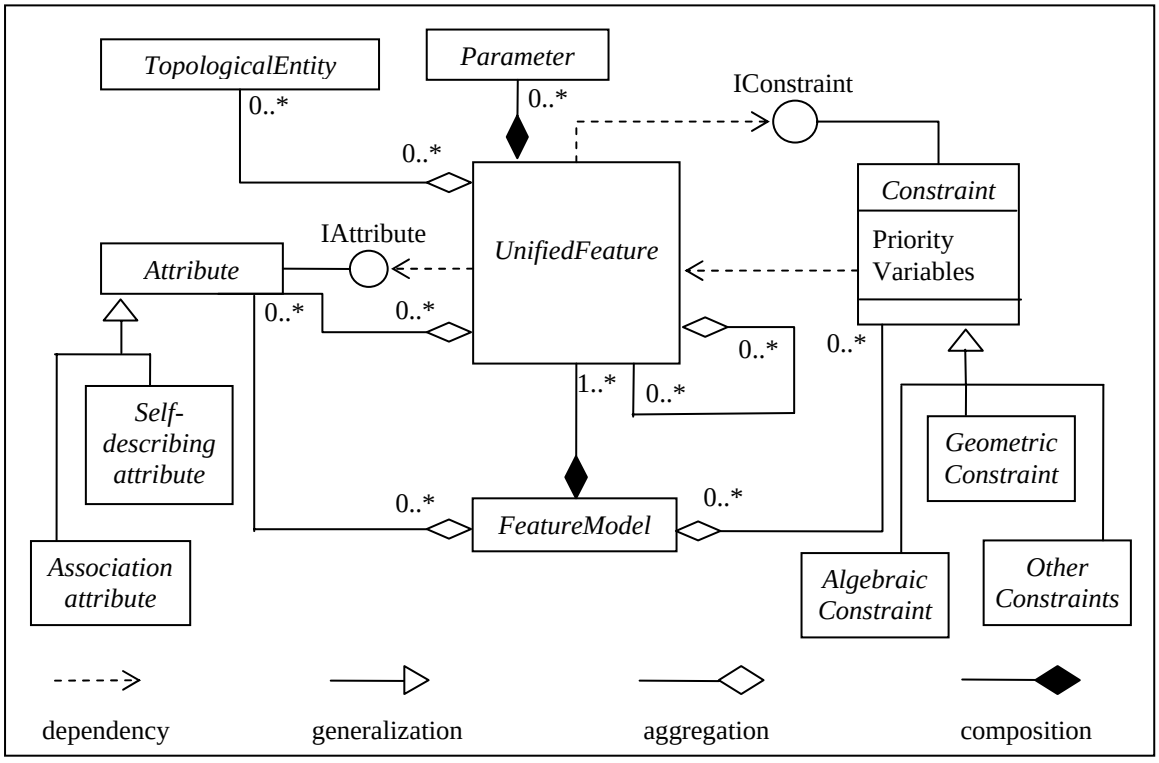


Figure 3.1 Unified feature

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 it depends on. 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 simple aggregation relationship. It describes strong ownership and coincident lifetime between the parts and the whole. The

ranges aside the origin and target of an aggregation (or composition) arrow indicate how many parts can or must be in a whole. 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. For the consistency of description, in some figures of this thesis, UML is used to describe classes, modules, and flow charts.

All other application features are derived from the unified feature class. They inherit generic fields and may override generic methods. Application features also define their specific fields and methods. The main fields and methods of the unified feature class are described as follows (also see Table 3.1).

3.2.1 Fields

The unified feature class has four main kinds of fields:

- *Non-geometric attributes* represent feature properties that are attached to the feature or to the feature's geometric entities. They do not directly describe a feature's shape. Attributes are further classified into *self-describing attributes* and *association attributes*. Self-describing attributes represent properties, which are special to a particular feature class. Examples of self-describing attributes are material type, surface finish, and feature nature (adding or removing material). Association attributes are references to the entities associated to this feature, such as other features, corresponding facts in the expert system, etc. In addition, association attributes are used to refer to non-geometric entities. For example, they refer to functions and behaviors in the conceptual design stage, or machine tools and machining operations in the process planning stage.
- *Geometric parameters* describe a feature's geometric shape, dimension, position, and orientation, such as the origin position and length, width, height of a block feature. Geometric parameters are used as input to the geometry creation methods provided by the geometric modeling kernel.
- *Constraints* can be classified according to the elements they constrain:

(i) Intra-feature constraints restrict the field values in a feature. For example, a pocket's width equals to its length or a blind hole's bottom face must be on the part boundary.

(ii) Inter-feature constraints specify relations between two or more features.

(iii) Constraints can also be specified between a feature and other entities. For example, a process planning rule is used as the constraint to specify whether a cutter can be used to create a feature with the specified shape, dimension, tolerance, and surface finish.

Constraints can also be classified according to their types. Three types of constraints are supported in the proposed scheme:

(i) Algebraic constraint.

(ii) Geometric constraint.

(iii) Rule-based constraint, which are used to restrict a feature's presence or the values of feature properties directly based on engineering rules. Details are given in Chapter 4.

Constraints have priorities.

- *Geometric references* are pointers to topological entities in a geometric model. Since features are used to describe specific relations between topological entities, a feature's geometry is not necessarily volumetric, connected, or two-manifold.

3.2.2 Methods

Interfacing functions, which deal with geometric modeler, knowledge engineering module, relation manager, and database, are defined in the unified feature class. The overall structure of the required modules for implementing this proposed unified feature scheme is explained in Section 3.6.

Creating and Editing Feature Geometry

In the proposed scheme, conceptual design and detailed design features are created from predefined and parameterized geometric templates. In the process planning stage, with a design feature model as input, a process planning application analyzes all machined faces for suitable process planning features. The properties of these faces are then used to determine the values of process planning features parameters (details in Chapter 5). Feature parameters are used to create product geometry with the help of

functions provided by a geometric modeler. The way geometry is created is delegated to the specific features.

Feature geometries can be 2D faces or 3D solids in the developed scheme. The geometries of different dimensional features are represented uniformly in a geometric model (Chapter 4). When an application feature is created, its geometry is inserted into the geometric model. When a feature is modified, it notifies the geometric modeler about the modifications. In both cases, the geometric model will be updated accordingly.

Supporting Knowledge Embedment

When an application feature is created, a corresponding fact is generated and inserted into a knowledge base. The fact describes the feature's identity, its parameters, and self-describing attributes. The fact generation and insertion methods are defined in the unified feature class. When a feature is altered, it notifies the knowledge base.

Supporting Data Associations and Validity Maintenance

In a single stage, when an application feature is created, a corresponding node is generated and inserted into a relation manager. The relation manager is responsible for managing the dependency relations among entities. The constraints, which are responsible for the feature's presence or for controlling the values of feature parameters or self-describing attributes, are also inserted into the relation manager and are associated to the corresponding feature nodes (details in Chapter 6). The node generation, insertion, and association methods are defined in the unified feature class. When a feature is modified, it calls the relation manager for change propagation. Related constraints are validated.

To support inter-stage data sharing, associations, and change propagation, application features as well as their inter-relations are stored in a common database. The methods of storing features into the database are defined in the unified feature class.

Two points about the unified feature definition are worth noting:

- Traditionally, numerical constraints are used to represent engineering intent. As an extension, the unified feature class also defines associations to knowledge base, geometric model, and other non-geometric entities in order to represent and maintain engineering intent.

- From the viewpoint of software engineering, data sharing is difficult because one application does not know the data structures of other applications. Hence, applications cannot manipulate the data created by other applications. With the unified feature definition, feature data sharing among applications is improved on:
  (i) An application feature may have its specific properties, which are not included in the unified feature definition. However, with both application features defined as sub-classes of the unified feature class, an application understands the generic part of feature objects of other applications. The generic data is then used to reconstruct unified feature objects (Figure 3.2).
  (ii) In the proposed scheme, each application stores the data in a central relational database. An application can access the database to retrieve the data that is authorized.

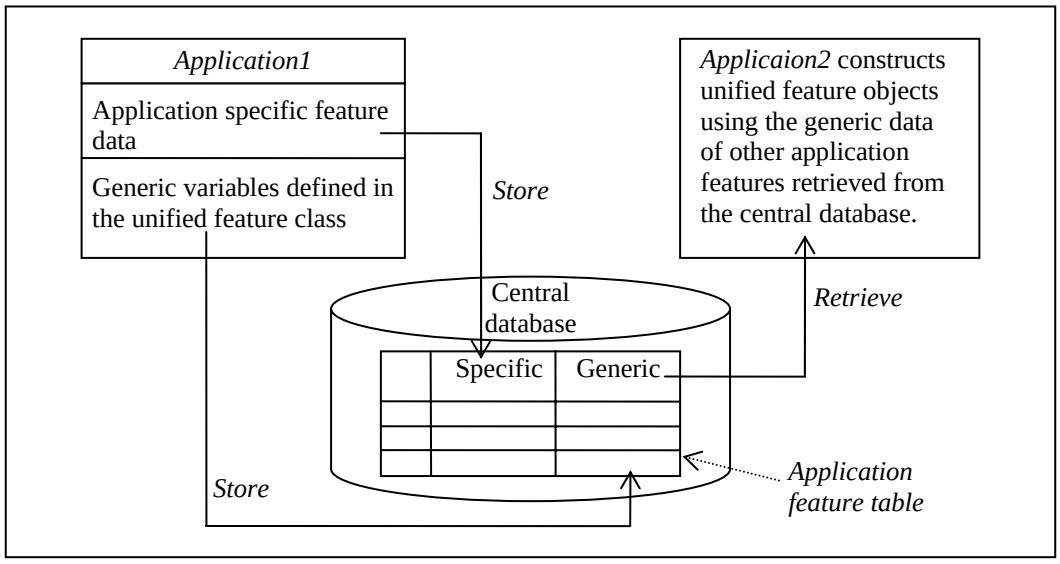


Figure 3.2 Data access methods via generic fields of application features

Table 3.1 describes the major fields and methods defined in the unified feature class.

Table 3.1 Major fields and methods of the unified feature class

	Name		Description
Fields	Attributes	Association attributes	Identifiers of the associated objects, such as functions and behaviors in a conceptual design, machines and cutters in a process plan, other features, etc.
		Self-describing attributes	Material, surface finish, belonging application, etc.

Table 3.1 Major fields and methods of the unified feature class (continued)

	Name		Description
Fields	Parameters		Variables used as input to geometry creation methods
	Constraints	Geometric constraints	Identifiers of geometric constraints that the feature's topological entities participate in
		Algebraic constraints	Identifiers of algebraic constraints that the feature's self-describing attributes or parameters participate in
		Rule-based constraints	Identifiers of rules that the feature or its self-describing attributes, parameters, or numerical constraints participate in
	Geometric references		Topological entities
Methods	Geometry construction	createGeometry()	Generate the feature geometry
	Interface to geometric modeler	getCell()	Find out the feature's member topological entities
		setCell()	Assign a topological entity the feature's identifier
		insertGeometry()	Ask the geometric modeler to insert the feature geometry
		deleteGeometry()	Ask the geometric modeler to delete the feature geometry
	Interface to expert system	getFact(), setFact()	Retrieve or create the corresponding facts
		getRule(), setRule()	Retrieve or assign the corresponding rules
		checkRule()	Check whether the related rules are satisfied or not
	Interface to relation manager	addToJTMS()	Add the feature or its self-describing attributes, parameters to the relation manager as nodes
		validityChecking()	Call the relation manager for feature validation
	Interface to database	storeFeature(), retrieveFeature()	Store a feature in or retrieve a feature from the database

3.3 The Product Model

The elements of the proposed product model and the manner these elements are defined are described as follows.

3.3.1 What should be Included in a Product Model?

Traditional product model, such as product designs or process plans, usually contains only the final output of reasoning processes. For example, in design stage, only product shape and specifications are recorded. Similarly, in process planning, only processes used (covering machine tools, machining conditions, and machining sequence) are recorded. The engineering intent or reasoning processes behind such results is usually not represented and recorded explicitly. The proposed scheme represents engineering intent explicitly but collectively in a product application model as associations among engineering knowledge, non-geometric entities, product structures and specifications

(Figure 3.3). The engineering knowledge is represented as application specific rules, such as function-to-behavior mapping rules or machining method selection rules.

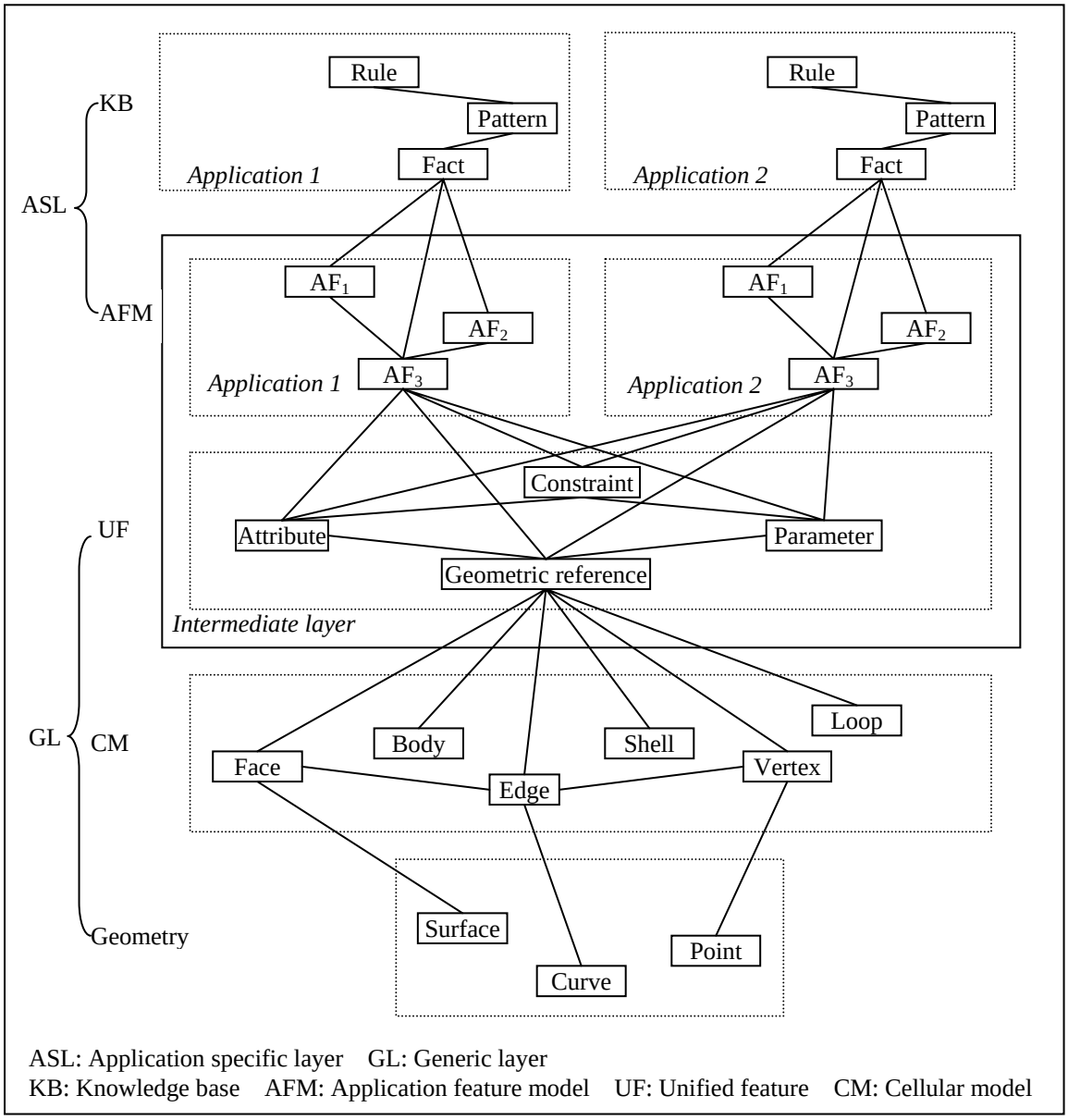


Figure 3.3 Data associations in the unified feature-based product modeling scheme

Figure 3.3 illustrates the hierarchical structure of the proposed product model as well as the associations among data. In the figure, the application specific layer is concerned with data that is specific to particular computer aided tools. For example, a conceptual design knowledge base, product functions, and abstract conceptual design features are defined only in the conceptual design application while the machining operation elements and setups are defined only in the process planning application. Generic

entities, which include attributes, parameters, numerical constraints, cellular models, JTMS, unified features, are used with no discretion to applications. Entities in the application specific layer, such as application features, can consist of or refer to generic entities. Generic entities can be instantiated and manipulated by application specific entities.

Due to the different natures of knowledge model and geometric model, an intermediate information layer is needed. The application and unified feature models represent interfaces upward to the knowledge model and downward to the geometric model. The unified feature model also describes associations between features. These features may belong to the same or different development stages. As illustrated in Figure 3.3, features connect to each other in several ways: through common topological entities, constraints, and so on.

The proposed product model comprises the following elements:

- *Engineering knowledge*, which is represented and manipulated using a rule-based expert system.
- *Reasoning processes* deduce from a set of input using clearly defined complex algorithms. Examples of reasoning processes include function decomposition or setup minimization processes.
- *Entities* can be geometric or non-geometric. Non-geometric entities, such as functions and behaviors in product design; machines and cutters in process plan, are associated with product geometry via features.
- *Data relations* are constraints, which determine the scope of values of related entities. Three types of constraints (algebraic, geometric, and rule-based) are supported in this scheme. These constraints will be explained in Chapter 4.

3.3.2 What should be Transferred among Applications?

As mentioned in Chapter 1, product development stages are inter-related and mutually constraining. Therefore, to keep the consistency of data throughout these stages, some of the product data and their dependency relations or associations need to be shared. This requirement is usually hampered by incompatible or proprietary data structures. In the previous sub-section, product model elements are identified. Application specific contents need not to be defined commonly if they need not to be

shared. However, geometry and features should be shared because a common product master model is constantly referred and hence must support common definitions.

Although for geometric and topological data, IGES and STEP (such as [27]) are widely supported by mainstream CAx systems, but for feature data, as reviewed in Chapter 2, there is no commonly accepted definition yet. The proposed unified feature concept is intended to support sharing feature data among conceptual design, detailed design, and process planning stages. Note that this unified feature concept has enriched traditional feature concepts for the characteristic of data associations.

3.4 Data Association Mechanisms

The data within and among applications must be associated for validity maintenance.

3.4.1 Intra-Application Data Association Mechanism

In a single application, product data are associated using a relation manager and a geometric modeler. The relation manager is used to record the dependency relations among entities, such as rules, numerical constraints, non-geometric entities, and features (Chapter 6). The geometric modeler is used to share topological entities among features (Chapter 4).

3.4.2 Inter-Application Data Association Mechanism

In the proposed scheme, applications are independent and separate. They use a central database to share and associate geometric and feature data. The inter-application data associations are established in two levels (see Figure 3.4). In this figure, solid arrowed lines represent data flow for recording and sharing.

- Geometry-level associations. All three life cycle stages relate to the final product's shape. For example, the conceptual design defines the skeleton of the final shape, while the process plan is a re-explanation of the final shape from the viewpoint of machining. Therefore, it is necessary to establish geometric associations among these three stages. The proposed scheme uses a non-manifold geometric modeler to establish and maintain geometric associations. Further details are given in Chapter 4.
- Feature-level associations. Besides geometric relations, non-geometric relations exist and must be established and maintained among applications. For example, a conceptual design feature (relating to functions, behaviors, states, and physical laws)

is transformed into a set of detailed design features (with specific shapes, dimensions, materials, surface finishes, etc.), which are further transformed to a set of process planning features (machines, cutters, machining sequences, volumes or faces). The corresponding relations are important in the propagation and validation of engineering intent. However, it is difficult to represent these relations using geometric associations or numerical constraints. Feature-level associations provide a more flexible and general way to maintain these relations.

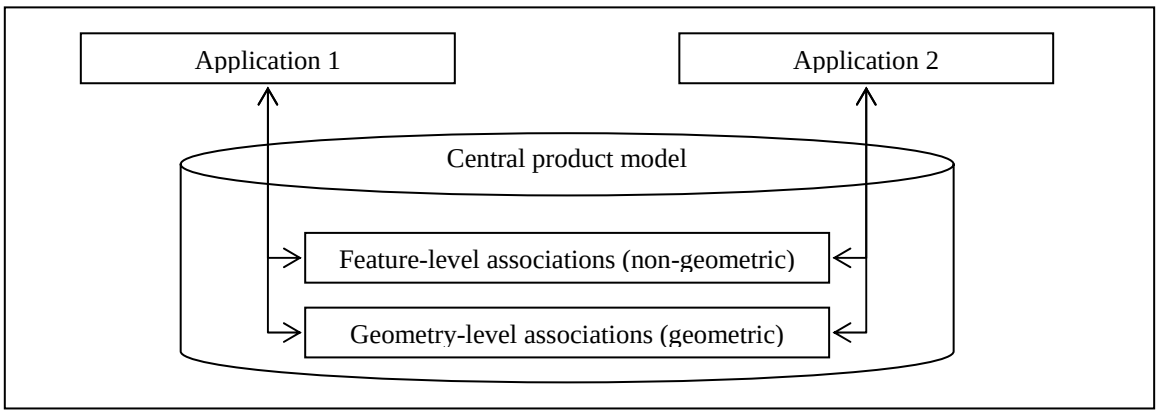


Figure 3.4 Inter-application data associations using a central database

Figure 3.4 shows data sharing across applications based on a central database. The purpose of the inter-application data sharing mechanism is:

- To re-establish the relations between geometric and non-geometric entities of the other application even without the specific application feature definitions through re-constructing unified feature objects (and hence the association attributes and geometric references, see Section 3.2). This re-establishment is critical to understanding the embedded engineering intent of other applications.
- To propagate modifications among applications where inter-application associations will be used.

3.5 Change Propagation Algorithm

With the product model and the established associations, modifications are propagated within and among applications.

In each application operation, the application stores data into the appropriate tables of the central database according to data types. When inter-application associations are

created, they are stored into association tables in the central database as well, i.e. association tables for features and geometric entities respectively.

When modifications have been requested by an application, they are validated within the application first. Then the central database is notified about the locally-approved modifications. Based on the stored associations, the central database notifies the affected applications. Each affected application checks the validity of the modification. If the local validation cannot be passed, the application feeds back the central database to reject the modification in order to maintain the inter-application consistency. If all applications approve the modifications, the central database will be updated. Further details are given in Chapter 6.

3.6 The Structure of the Unified Feature-Based Product Modeling Scheme

The structure of the unified feature-based product modeling scheme is briefly described here (Figure 3.5). The implementation methods and the prototype system will be introduced in Chapter 7. In Figure 3.5, solid lines represent data reference relations while dashed lines represent inheritance relations. For example, application features are sub-classes of unified features defined in the unified feature modeler.

Major functional modules are:

- A central database maintains inter-stage associations.
- A relation manager maintains local dependency relations.
- A geometric modeler provides geometric modeling functions. It also accommodates different geometric modeling requirements, maintains geometric associations among features, and keeps the canonical shape of features.
- A rule-based expert system represents and manipulates engineering knowledge.
- A numerical constraint solver evaluates and solves geometric and algebraic constraints.
- A unified feature modeler defines the fields and methods of the unified feature, which are common to all application features.

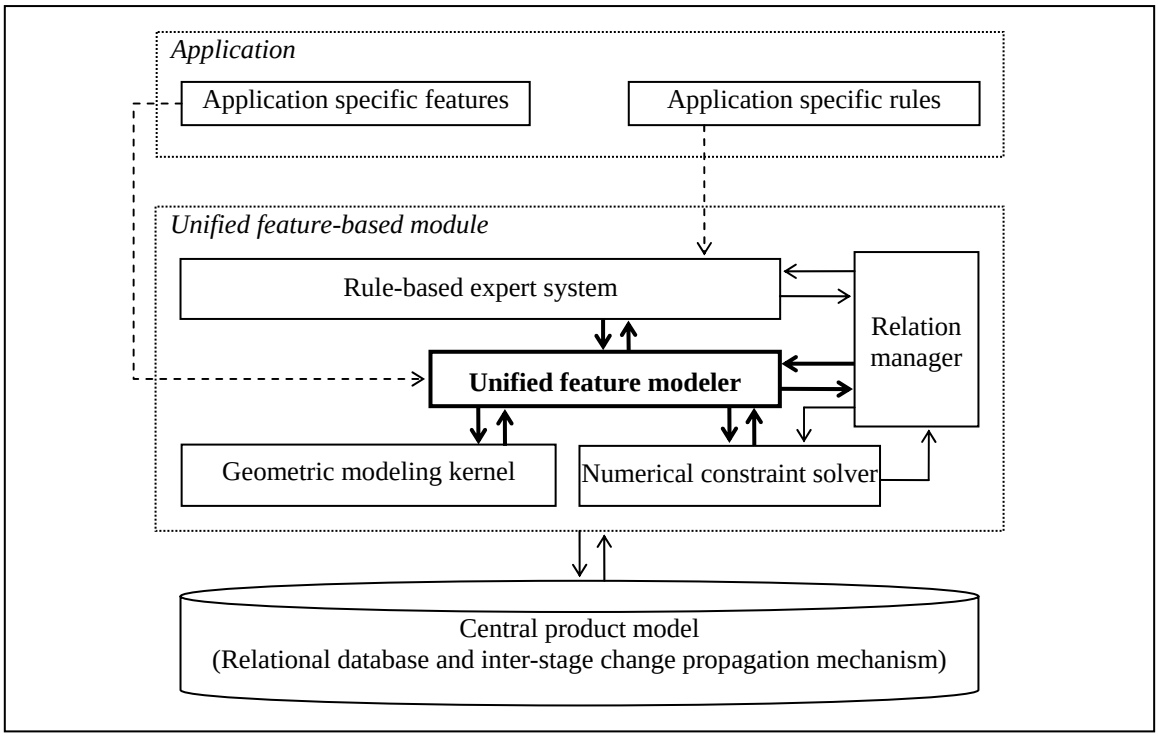


Figure 3.5 Unified feature-based product model

3.7 Summary

The unification of applications modeling product life cycle stages is achieved in the proposed scheme as follows:

- The unified feature specifies common fields and methods of multiple application features.
- The unified geometric representation accommodates distinct geometric modeling requirements of multiple applications by allowing both non-manifold and two-manifold geometric entities and linking them persistently.
- Common mechanisms embedded in modules are developed, such as embedding knowledge into product models, associating application features via the generic fields of the unified feature, and change propagation methods.
- The way of maintaining the validity and consistency of product models is generic.

With the unified feature scheme (originally proposed in [Chen et al. [180]), in each application, vertical data associations are established between engineering knowledge and product geometry via features. Across applications, horizontal data associations, especially non-geometric associations are established also based on features.

Two major extensions of the traditional feature technology in this scheme are the embedded knowledge model and the non-manifold geometric model; they are described in the next chapter. Furthermore, the definitions of conceptual design features and process planning features are given in Chapter 5.

Chapter 4

Knowledge Embedment and Cellular Topology

4.1 Introduction

The unified feature-based product modeling scheme extends traditional feature technologies by integrating them with a knowledge-based reasoning system as well as a non-manifold geometric modeler. In this chapter, the knowledge embedment and the multi-dimensional cellular topology are introduced.

Generally, the information model for each product development stage has four main models, which are knowledge, application feature, unified feature, and geometric models:

- The knowledge-based semantic model comprises abstract engineering knowledge (in the form of rules in this work), existing facts, and reasoning processes (Giarratano & Riley [181], Hill [182], Bigus & Bigus [183]). The inference engine accesses the knowledge base to find rules that are ready to fire. The fired rules insert new facts to the knowledge base or invoke feature methods.
- The geometric model represents product geometry and topology.
- The geometric modeling systems are inherently restricted to geometric information. As the intermediate layers between knowledge and geometry, feature models are necessary to associate non-geometric information, in the form of attributes, parameters, and constraints, to the geometric model.

4.2 Knowledge Embedment

According to [181], the appropriate domain for a knowledge-based system (expert system in particular) has the following characteristics:

- Ill-structured problems. Usually, there is no efficient algorithmic solution.
- The problem solving knowledge is mainly heuristic and uncertain.

Development stages of a mechanical product exhibit the above characteristics and hence are appropriate application domains for a knowledge-based system. In fact, engineering knowledge has already been ubiquitously used in the product development processes. Figure 4.1 and 4.2 illustrate the engineering knowledge used in the design and process planning stages.

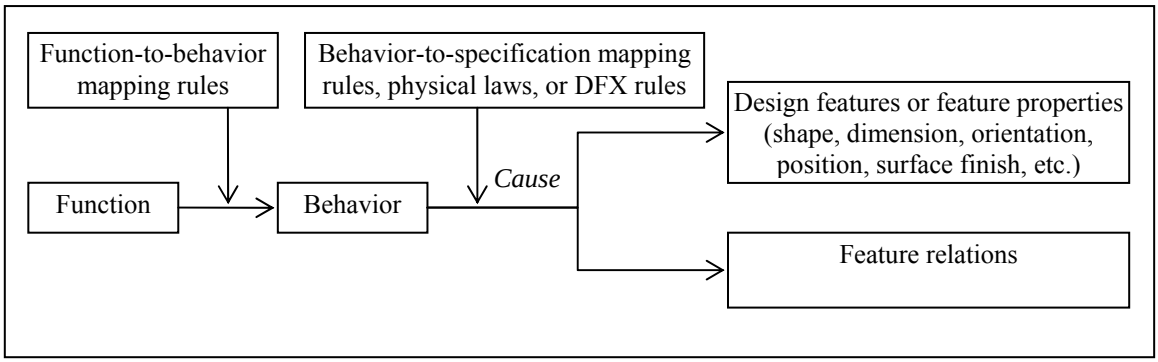


Figure 4.1 Engineering knowledge in design stage

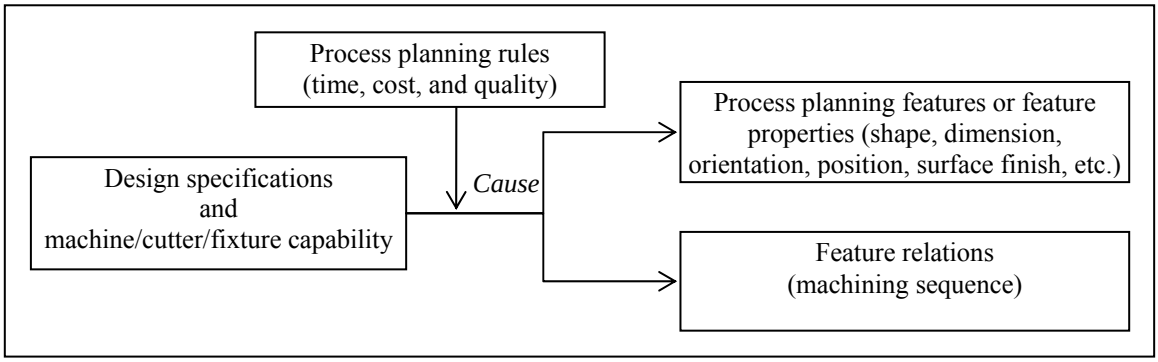


Figure 4.2 Engineering knowledge in process planning stage

However, as indicated in Chapter 2, previous research efforts in using knowledge in CAx systems have limitations:

- Three issues, which are how to embed engineering intent into product models using KBE, how to use engineering knowledge to drive product modeling or process planning, and how to associate engineering knowledge with product designs or process plans, have not been fully investigated.
- The information flow between the KBE system and the CAx systems are usually one-way from the KBE to the CAx systems. Using knowledge to justify modifications, to provide explanation for decisions, and to search for alternative solutions, is difficult.

The unified feature-based product modeling scheme integrates the KBE systems and the feature-based CAx systems in a flexible and more complete manner.

4.2.1 Integration of Knowledge, Feature, and Geometric Models

In this research, a rule-based expert system is used to represent and manipulate engineering knowledge. A rule-based expert system consists of three major parts: a rule base that stores rules, working memory that records current facts, and an inference engine that prioritizes and executes the rules (Figure 4.3). On one hand, the embedded expert system can invoke feature methods directly, such as creating or editing features. On the other hand, feature objects and their properties, if they are involved in the rule-based reasoning processes, have their corresponding facts created and stored in the knowledge base. A knowledge base does not interact with the geometric model directly. Instead, the feature model is used as a medium to bridge the gap between engineering knowledge and product geometry.

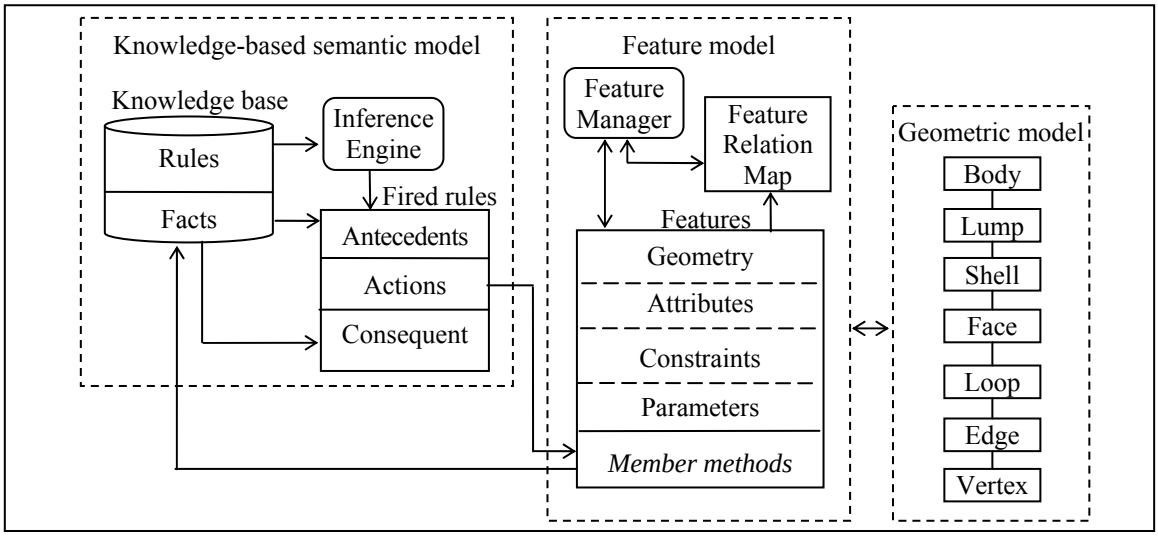


Figure 4.3 Relations between the knowledge model and the feature model

A knowledge base can be represented as the sum of facts F and rules R. A rule has two parts: the antecedents that must be satisfied before the rule can be fired, and the consequents that are generated (new facts) or executed (actions) when the rule fires:

$$F_{A1} \wedge F_{A2} \wedge \ldots \wedge F_{Ai} \rightarrow \left(F_{C1} \wedge F_{C2} \wedge \ldots \wedge F_{Cj}\right) \vee \left(F'_{C1} \wedge F'_{C2} \wedge \ldots \wedge F'_{Ck}\right) \vee \ldots \vee \left(F''_{C1} \wedge F''_{C2} \wedge \ldots \wedge F''_{Cl}\right)$$

in which F_A represents antecedent facts and F_C represents consequent facts. In addition, F'_C and F''_C represent the alterative solutions.

Figure 4.4 uses a UML graph to describe major classes in a forward-chaining rule-based expert system. A rule-based system usually uses pattern matching in its reasoning process. That means the antecedent parts of a rule are patterns, which consist of facts and the corresponding specified values or value ranges. When the current values of these facts match with the specified values or value ranges, the antecedent pattern is satisfied and the rule is ready to fire. The result of firing a rule may be asserting new facts or modifying old facts (specified in the consequent part of the rule). Firing a rule may also invoke actions and execute methods. Actions can also be used in the antecedent part of a rule to enquire other modules (such as the feature model in Figure 4.3) for necessary information.

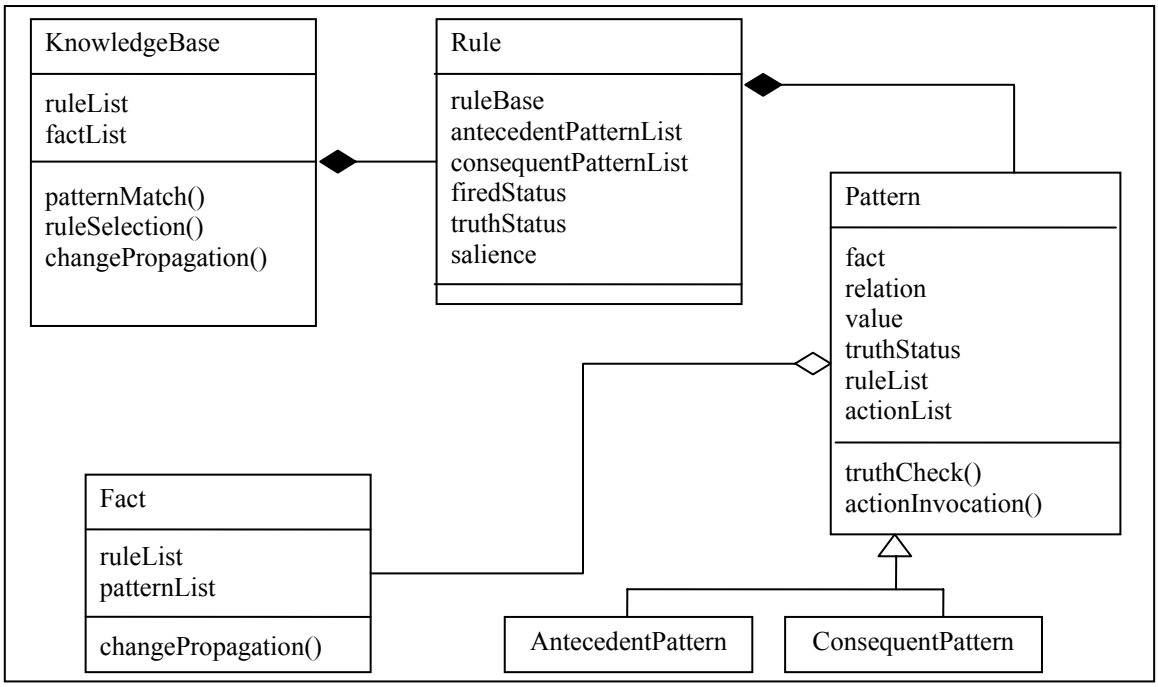


Figure 4.4 A forward-chaining rule-based expert system

Major entities in the knowledge model include rules, patterns, and facts (Figure 4.4). The major variables and methods of these classes are:

- The *KnowledgeBase* class stores rules and current valid facts. When new rules or facts are generated, they are inserted into it. The rule and fact lists are kept for pattern matching process and change propagation process. A pattern matching method is used to decide which rules are ready to fire by checking their antecedent patterns. Then the rule selection method selects one of the matched rules to fire.

Many strategies to select the most appropriate rule from the matched rule lists exist. As it is not the focus of this research, the first matched rule is simply selected to fire. The salience of a rule can be specified explicitly to affect this selection process. When facts are modified due to rule firing or feature modeling events, all the rules in the rule list are checked. The main actions undertaken by the expert system are:

(i) All rules that fired earlier with consequents referring to this modified fact are checked to ensure that the modification is in compliance with the rules. If not, the modification is rejected or the conflictions need to be solved.

(ii) All rules with antecedents referring to this modified fact are checked to see whether they are ready to fire.

This process is continued until all affected rules are checked.

- The *Rule* class keeps lists of antecedent patterns and consequent patterns. A rule can have multiple antecedent patterns or consequent patterns. In the developed rule-based system, all antecedents must be satisfied simultaneously before a rule is triggered. The relations between the consequent patterns may be conjunction or disjunction. The disjunction relations between the consequent patterns represent the alternative solutions. The *Rule* class uses a truth status to record whether it is satisfied under the current reasoning context. The fired status is used to record whether the rule has already been fired to prevent repeated firing. The salience variable is used to specify the rule's priority.
- The *Pattern* class comprises a *Fact*, a condition, and a specified value. When the current value of the fact satisfies the condition according to the specified value, the truth status of the pattern is set to true.

Penoyer et al. compared two options to integrate a KBE system to a CAx system: embedding the KBE system as an integral part of the CAx system, or separating them and using an application programming interface (API) for communication [177]. They expected the latter option to be more realistic for commercial software development. In this research, the former option is chosen in the aim of tightly integrating the KBE and the CAx systems and to reduce the programming overhead in defining the API.

4.2.2 Relations between the Knowledge Space and the Feature Space

Relations between the knowledge space and the feature space can be established in several ways:

- At all times, feature objects, constraints, attributes, parameters, and other application specific entities, if they are involved in the rule-based reasoning processes, have their corresponding facts stored in the knowledge base (Figure 4.5).

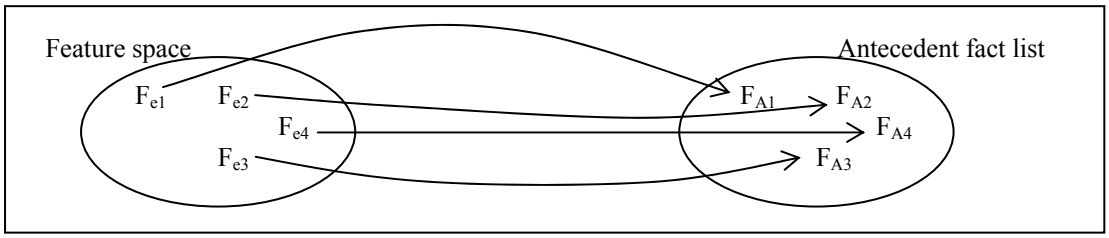


Figure 4.5 Features or their properties map to the antecedent facts

- The consequences of firing a rule may assert new facts or modify existing facts. There are two possibilities:

  (i) If the new or modified facts have fixed values, such as feature types, constraint types, or values of feature attributes, then these fixed values are used directly by the corresponding feature methods, such as creating or editing features (Figure 4.6).

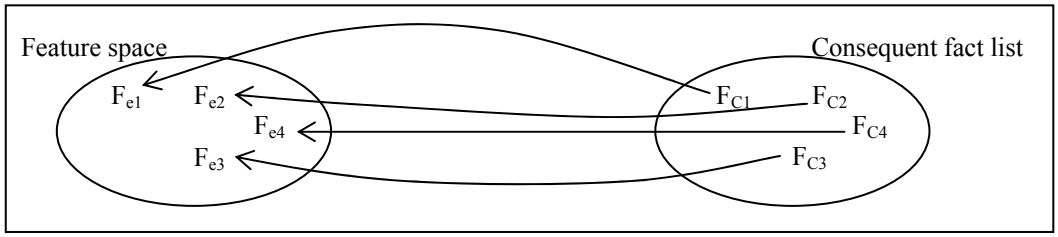


Figure 4.6 Consequent facts map to the features or their properties (one-to-one)

  (ii) If the consequent facts represent a range of values (continuous or discrete, circles in Figure 4.7), a valid default value is selected and the corresponding feature methods are invoked. Other allowable values are also recorded for a later exploration.

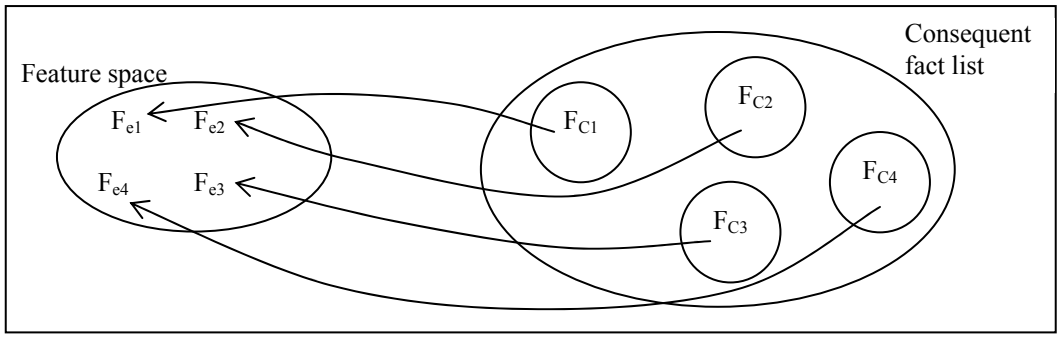


Figure 4.7 Consequent facts (value range) map to the features or their properties

When a rule fires, the corresponding nodes are added to the Justification-based Truth Maintenance System (JTMS) dependency network. The JTMS and its application in the unified feature-based product modeling scheme are introduced in Chapter 6.

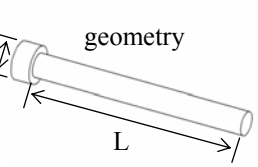
4.2.3 Rule-Based Constraint and Its Solving Mechanism

In general, a constraint restricts the values that the involved variables can assume simultaneously.

In the domain of mechanical CAD, constraint-based design approaches are commonly used to maintain the integrity of a product model during design modifications. Many constraint solving mechanisms and algorithms have been developed (Lottaz et al. [184], [119], Laakko & Mantyla [185], Bouma et al. [186], Tsang [187], Sannella [188], Kramer [189], Rossignac [190], Light & Gossard [191]). Most of them concentrate on representing and handling algebraic or geometric constraints. These numerical constraints determine low-level product specifications, such as the values of detailed product attributes or parameters, e.g. positions, orientations, and dimensions. Usually they are applied only in the detailed design stage.

By comparison, high-level product specifications or process plans are usually determined by functional requirements or manufacturing requirements. For example, a component type or a product layout depends on a particular design function, or a machining operation depends on the surface finish requirements. This kind of constraints handles different types of variables or entities than geometry or parameters (Table 4.1).

Table 4.1 Comparing numerical and rule-based constraints

Type	Constraint representation	Constraint mechanism
Parametric constraint	$D = 20$ $L = 100$	values of parameters D and L ⟷ *determine* / *two-way* 
Algebraic constraint	$b = a + 1$	value of a → *determine* / *one-way* → value of b
Geometric constraint	Line a is parallel to line b	direction of a ⟷ *determine* / *two-way* direction of b
Rule-based constraint	Fired rules: IF: "machining process" equals to "hole-making processes" AND "surface finish of a hole" is less than "RA5.0" AND "surface finish of a hole" is larger than "RA2.5" THEN: "machining route" equals to "rough boring and semi-finish boring"	value of surface type, surface finish, (antecedents) → *determine* / *one-way* → value of machining route (consequents)

They cannot (or at least cannot efficiently) be represented as numerical expressions and solved using numerical methods. Embedding knowledge-based systems into CAx systems provides a solution of handling this kind of constraints. In the unified feature-based product modeling scheme, these constraints are called *rule-based constraints*, since a rule-based expert system is used to represent and solve them.

Definition of Rule-Based Constraint

- A rule-based constraint consists of a set of enabling fired rules. The involved variables or entities can be geometric or non-geometric. The variables or entities are represented as facts. A rule-based constraint associates its antecedents and consequents. Some intermediate variables may be involved if more than one rule is fired.
- A rule-based constraint usually cannot (or should not, due to the complexity) be solved using mathematical methods. They are more appropriate to be solved using KBE approaches, such as forward-chaining reasoning. In contrast, relations between

member variables of a numerical constraint are usually represented as mathematical formulae.

- A rule-based constraint restricts the, possibly symbolic, values of their associated entities. Unlike numerical constraints (which can be directed or undirected), rule-based constraints are always directed from antecedents to consequents.
- A rule-based constraint is generated after a successful forward-chaining reasoning process.
- A rule-based constraint is said to be satisfied if, under the given state of the initial antecedents, all the fired chaining rules are satisfied and the ultimate consequents can be deducted accordingly as the results of the forward-chaining reasoning process.
- Rule-based constraints can generate numerical constraints, which are added to the product model (and retracted if the rule-based constraint is unsatisfied later).

Solving Rule-Based Constraints

When the properties of the antecedents or the consequents of an established rule-based constraint are modified, the rule-based constraint has to be validated. If the constraint is found to be violated, it needs to be resolved. Two situations may occur:

- The values of the initial antecedent variables are modified.
  (i) If the new value of the modified variable still satisfies the antecedent pattern, then no more checking is needed.
  (ii) If the new value of the modified variable does not satisfy the antecedent pattern, then the consequent state (including facts or numerical constraints, if any) is revoked. A new forward-chaining reasoning process (pattern matching, selecting, and firing rules) is invoked in aim of creating a new applicable constraint. If the process is successful, then the rule-based constraint is re-generated. Otherwise, the constraint is said to be violated.
- The values of the consequent variables are modified:
  (i) If the new value of the modified variable still satisfies the consequent pattern, the rule-based constraint is satisfied and no more checking is needed.
  (ii) If the new value of the modified variable violates the consequent state:
    a. If the modification is also the result of a rule-based reasoning process, it indicates that the two sets of rules (the one that causes the modification and

the presently evaluated one) conflict. The rule base needs to be revised or a human intervention is needed.

b. If the modification is not caused by a rule-based reasoning process, the modification is rejected or a human intervention is needed.

The rule-based constraints enable the unified feature-based product modeling scheme to justify feature modifications using rule-based reasoning mechanisms. As proposed in (Chen et al. [192]), if a feature or its properties are causes or results of a rule-based reasoning process, they are associated with the respective rules, facts, and rule-based constraints. Feature modifications need to be evaluated through checking the associated rules.

4.2.4 Justifications, Explanations, and Alternative Solutions Provided by KBE

With the mapping relations between the knowledge space and the feature space as well as the dependency network, the limitations of the embedded KBE systems mentioned in the beginning of this chapter can be addressed:

- All features and their properties, if they are involved in the rule-based reasoning processes, have corresponding facts in the knowledge space. All generated facts are linked to corresponding features or feature properties. Feature methods can be invoked directly from firing a rule. This seamless integration makes sure that the feature modeling process is driven and justified explicitly by engineering knowledge.
  - (i) Feature modifications are checked using the supporting justifications (if specified as rule-based constraints).
  - (ii) Backtracking during the searching process is dependency directed and not based on a chronological sequence. This is due to the justifications, i.e. the reasons for drawing a conclusion, are explicitly recorded with each node. This characteristic is also important in managing other types of associations in the unified feature-based product modeling scheme, which will be discussed later.
  - (iii) Explanations for a decision or the failure of a feature modification can be given because of the explicitly stored justifications.

In this way, engineering knowledge is represented and stored in the product model.

- With the embedment of the knowledge base, engineering knowledge is not necessarily transformed into algebraic or geometric constraints. Engineering knowledge can be stored in their natural representation (rule-based constraints) and justify the related features. For example, the realization of a product function may require the presence of several features. Algebraic or geometric constraints are unsuitable to represent this kind of relations. With rule-based constraints, whenever one of these features is modified, the dependency network allows finding all the related features and checking their validities.
- Since the consequence part of a rule may have several disjunctive patterns, it means that potentially many alternative solutions are represented. Furthermore, a rule's consequent pattern may specify a value range for a feature parameter, attribute, or even type. During the forward-chaining reasoning, the engineer may choose a more promising option or a particular value for further exploration. If the later reasoning process identifies this as a bad choice, then other options or other values may be chosen. In this way, more alternatives for design or process plan optimization can be explored.

The dependency network established using JTMS records associations between knowledge, features, and other non-geometric entities in the unified feature-based product modeling processes. The JTMS, including the algorithms for propagation of modifications, are described in Chapter 6.

4.3 Cellular Model

Traditional geometric modeling systems use boundary representation (B-rep) or constructive solid geometry (CSG) models for geometry representation [57]. They have limitations with respect to the requirements of the unified feature-based product modeling scheme.

- Only the final product geometry is stored and managed. Intermediate geometries, which do not belong to the final boundary, are usually not stored. This limitation makes feature modifications difficult. It also results in a persistent naming problem.
- CAx systems have different requirements on representing product geometry. A hybrid geometric modeling environment that can accommodate the associative

wireframe, surface, and solid models coherently is a natural outcome of the unified feature-based product modeling scheme.

- CAx systems need to represent the same product geometry in different ways. On one hand, geometry may be represented in different abstraction levels. For instance, a hole can be represented as a central line (plus a radius), a cylindrical face, or a cylinder in different contexts. On the other hand, product geometry may be represented in different ways. For instance, two adjacent faces in one application may be represented as a single face in another application. In addition, it is important for the unified feature-based product modeling scheme that higher level application features can use lower level topological entities to propagate modifications and control the information consistency. Relationships or constraints in higher levels (e.g. feature level) may also be specified using lower level (e.g. topological entity level) relations.

Some solutions were proposed to solve these problems. They include:

- Bidarra et al. proposed to use a cellular model to represent intermediate product geometry as well as to support links between different views ([163], [138]). However, their methods are confined to 3D features only.
- Other researchers proposed to use the multi-dimensional non-manifold topology (MD-NMT) to meet the geometric modeling requirements of different applications ([90], [162], [160] [161]). However, they did not fully apply the MD-NMT to the feature-based modeling processes.

It can be seen that a multi-dimensional geometric modeling environment, which is capable of propagating geometric modifications across feature models, does not exist.

4.3.1 Using Cellular Topology in Feature-Based Solid Modeling

The goal of using a cellular topology is to keep a complete description of all the input geometric entities without removing them after set operations on volumes (unite, intersect, and subtract), regardless whether they appear in the final boundary or not. The cellular model uses three mechanisms to fulfill this goal:

- *Attribute mechanism*. There are two kinds of attributes used in a cellular model:

(i) *Cell nature*. A cell is either additive or negative depending on whether it corresponds to materials of the product (or the topological entities on the part boundary) or not.

(ii) *Owner*. Each cell records its owing features because a cell may belong to several features due to feature interactions. The sequence of the owning features is kept to determine the cell nature.

- *Decomposition mechanism*. Two 3D cells do not overlap volumetrically. Whenever two cells overlap, new cells for the overlap are generated with the merged owner list.
- *Topology construction mechanism*. In a cellular topology based, non-manifold boundary representation, an operation on volumes does not remove any input geometry. The cellular model constructs topology (generates new faces, edges, and vertexes) before classifying the topological entities as *in*, *out* or *on the boundary*. All topological entities are marked and filtered for displaying according to the type of the operation.

Figure 4.8 describes a feature model based on a two-manifold boundary representation.

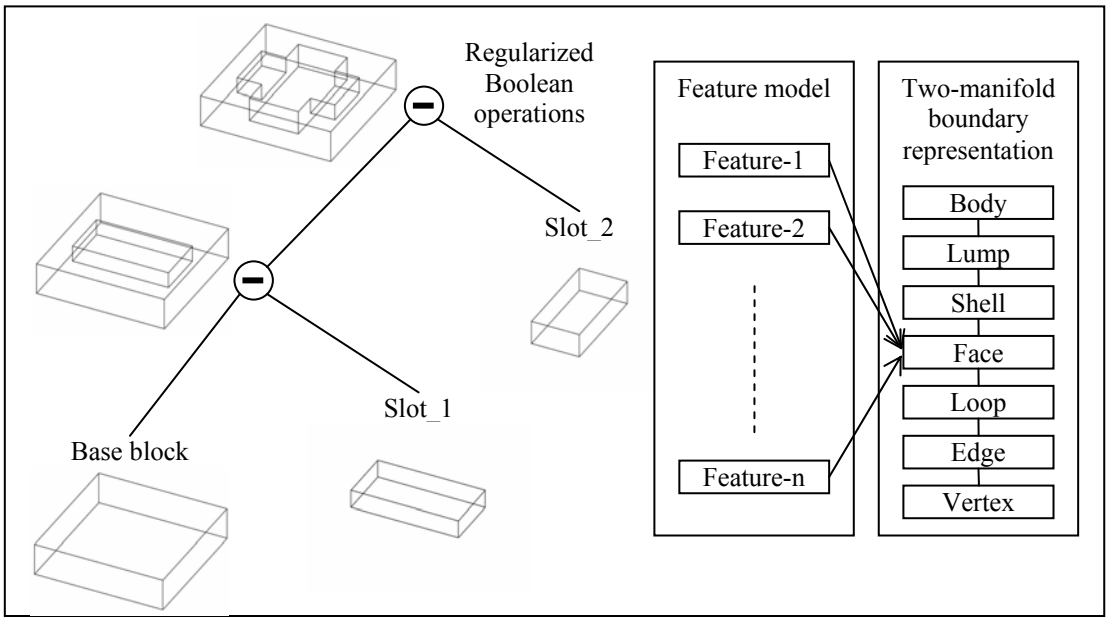


Figure 4.8 Traditional feature model based on a two-manifold boundary representation

Traditional boundary model does not store intermediate geometries. In other words, according to the types of Boolean operators, "useless" geometries are discarded before the part topology is reconstructed. For example, in Figure 4.8, when inserting the slot_1 feature, its top face is not stored. During the later modeling process, the intersections (due to feature interactions) further split and remove feature geometry from the boundary model. It is hence difficult to relate the feature to its corresponding topological entities in the final boundary model. In the constraint-based, parametric design processes, this limitation makes the feature model history-based. This limitation is also the major reason for the persistent naming problem [56]. Alternatively, in cellular topology based, non-manifold boundary representations, operations on volumes do not discard any input geometry. Part geometries are represented using cellular topologies. Figure 4.9 shows an example of the cellular topology based feature modeling process.

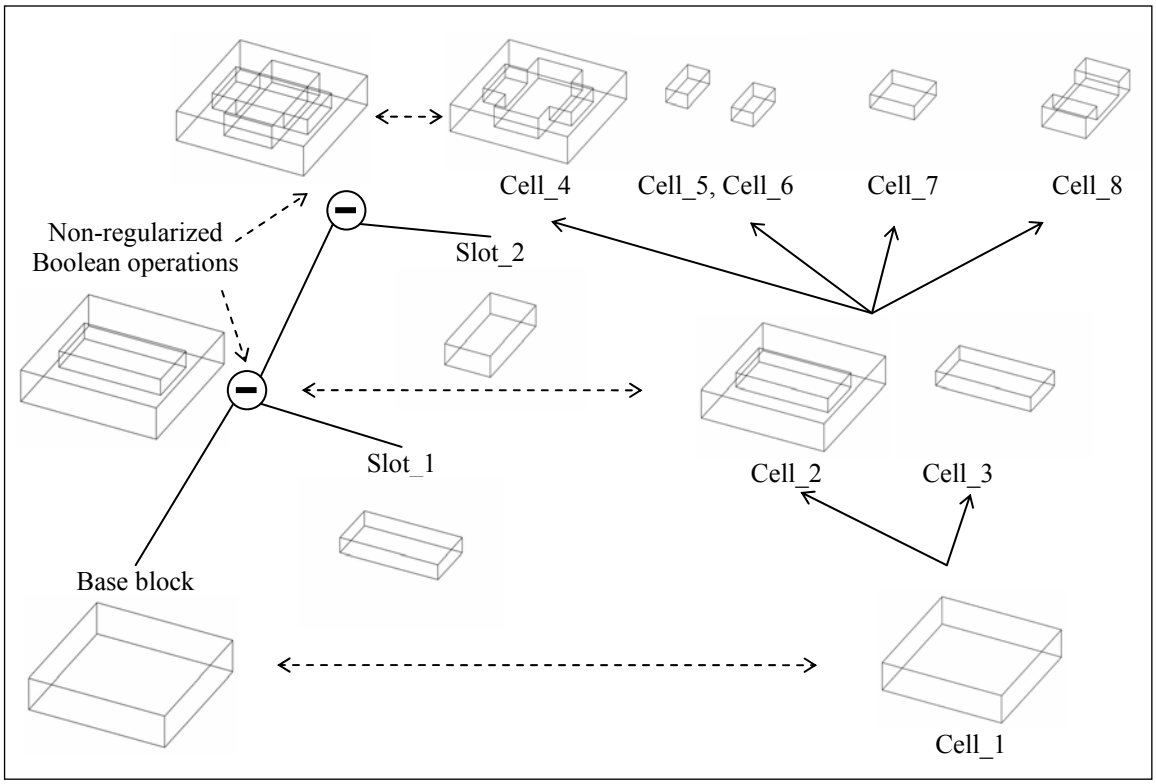


Figure 4.9 Feature model based on the cellular topology

The use of cellular topologies simplifies the representation of feature geometry as combinations of cells (Figure 4.10). When a feature is initialized, it is a single cell that carries only this feature's identifier. Whenever feature interactions occur, new cells that

belong to the intersections are generated. The newly generated cells carry a merged owning feature list.

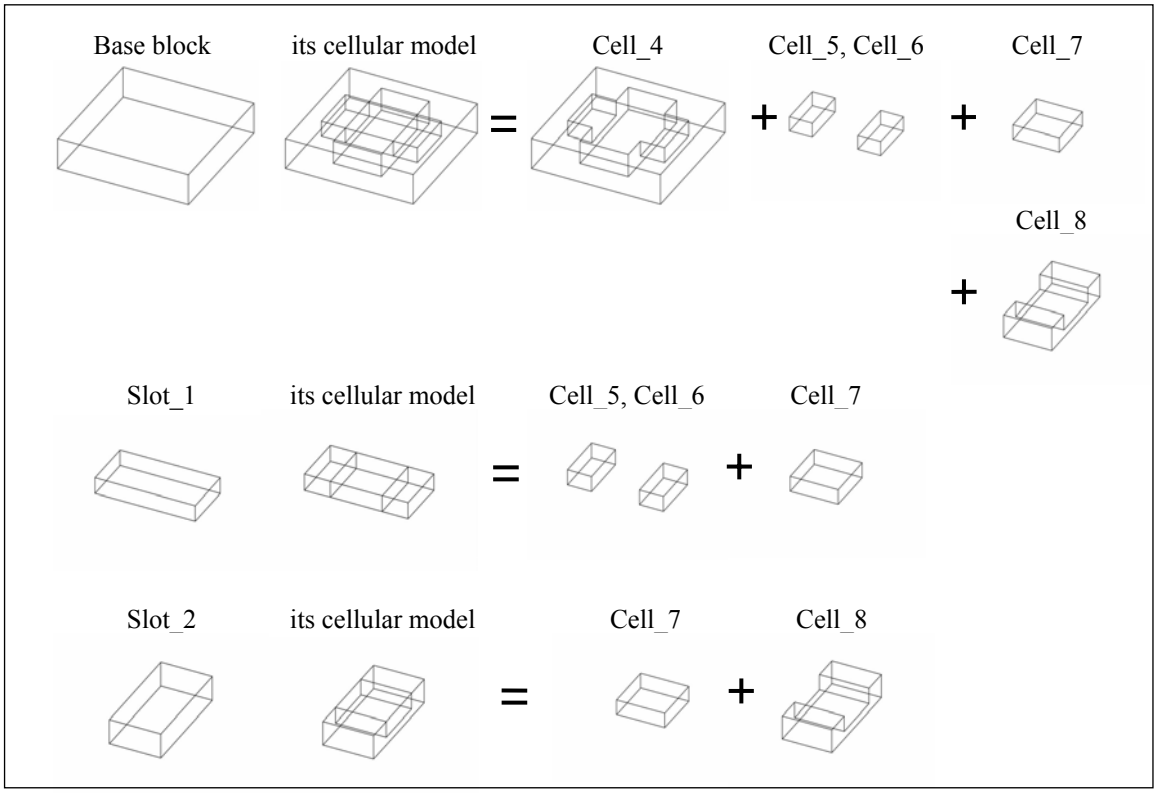


Figure 4.10 Feature geometries as different combinations of cells

The cellular model further links to the boundary model (Figure 4.11).

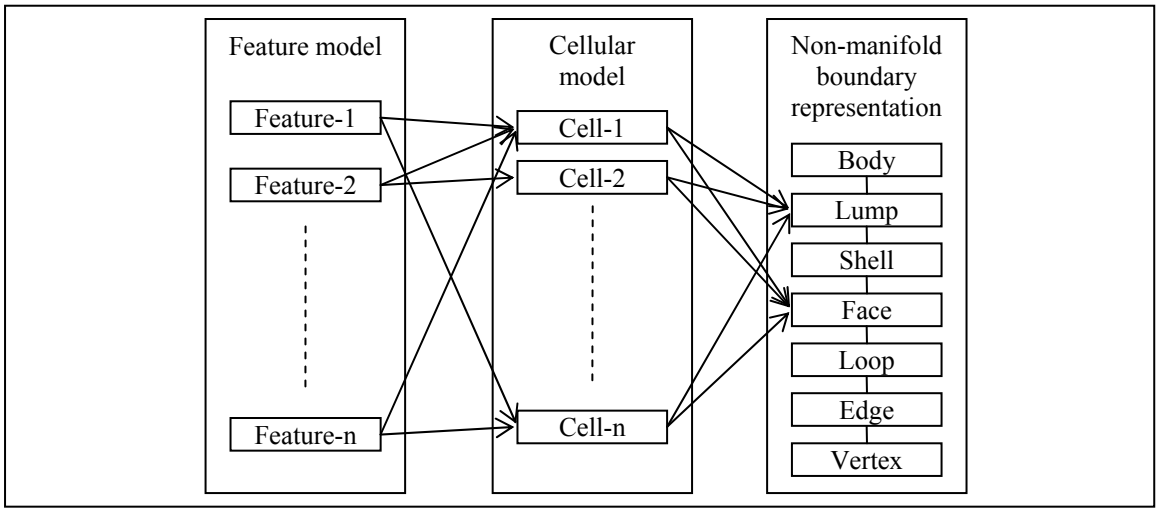


Figure 4.11 Feature model, cellular model, and non-manifold boundary representation

The geometry of a cell can describe any shape. For example, Figure 4.12(a) shows a block with three intersecting holes. Figure 4.12(b) shows the cell, which carries only the block as its owning feature. Figures 4.12 (c) to (e) show the cellular representations of the three hole features.

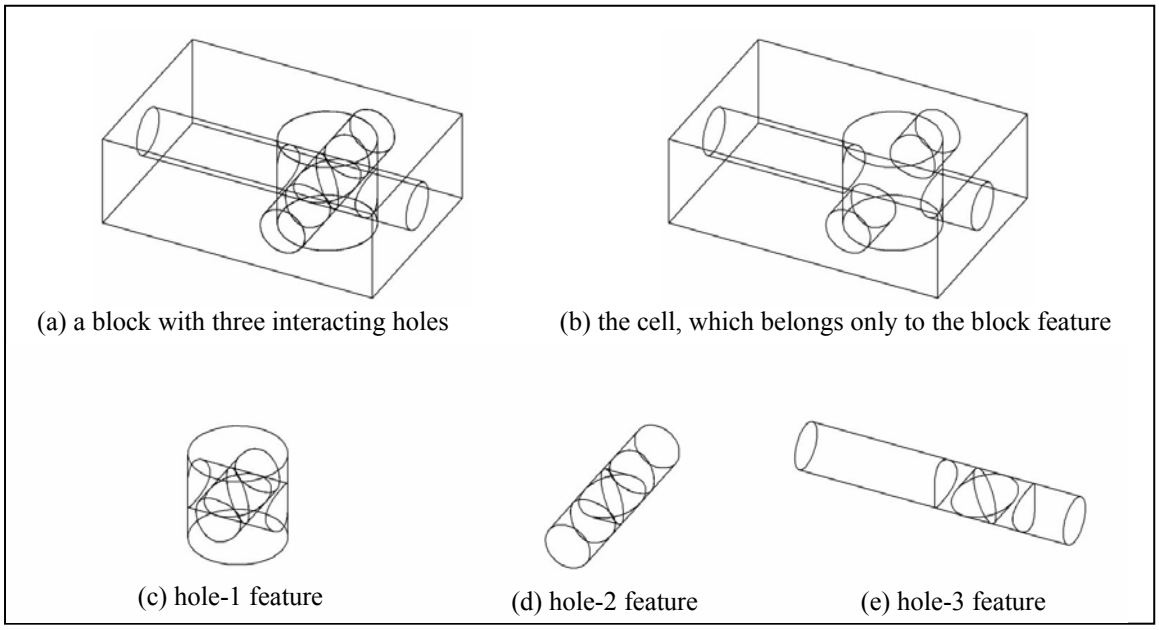


Figure 4.12 The geometry of a cell may have any shape

In this way, all feature geometry, geometric relations between features as well as relations between a feature and its corresponding topological entities are stored in the product model persistently. By using this approach, the persistent naming problem can be avoided. It is also possible to modify features based on the dependency relations, not on the construction history because the influence scope of a geometric modification is confined by the inter-cell relations. However, to meet the requirements of the unified feature-based product modeling scheme, this 3D-cell-based multiple-view feature modeling approach need to be extended.

4.3.2 The Extended Use of Cellular Model

Distinct applications covered by the unified feature-based product modeling scheme have their particular geometry representation requirements:

- During conceptual design, a designer is concerned about functions and behaviors. Only critical geometries and their relations are specified at that time. These critical

geometries may only be represented as abstracted lines, faces, curves, or surfaces. Solid models, detailed topologies and geometries are not specified in this stage.

- In the detailed design stage, the product geometries or layouts are further materialized. Two-manifold solid model representation is usually preferred.
- In the process planning stage, features are usually defined as material removal or accessing volumes related to machining operations. Fixtures are also conceptualized in this stage. For these types of features, solid representation with surface manipulation support is more appropriate because, other than the machined volumes, fixture design uses sub-area patches of the part, e.g. locating or clamping areas.
- Similar requirements are applicable to the assembly design stage. In particular, the sub-areas of the part or assembly for interfacing or grasping are concerned.

The geometrical representations discussed above relate to each other. They represent different aspects or abstraction levels of a product. To meet these diverse geometry representation requirements, the current cellular topology based feature modeling method needs to be extended to support not only 3D solid features, but also non-solid features. A multi-dimensional cellular model, named as *unified cellular model*, is proposed here to integrate all these representations, manage their relations and hence support the multiple-view feature-based modeling processes. The geometric model of each application is a particular aspect (a sub-model) of the unified cellular model. The traditional usage of the cellular topology in multiple-view feature modeling is extended in three aspects:

- 2D and 3D features are supported uniformly.
- The unified cellular model is used to share geometric data as well as to propagate geometric modifications (creating new cells, modifying or deleting cells) among views through the cells' owner attributes.
- Relationships in the cell level are generalized. These relation types can be used as building blocks to establish higher level feature relations.

The unified cellular model ensures the geometric consistency between application feature models:

- The geometries of a detailed design and the corresponding process planning model may have different topologies. However, both models correspond to the same final

product geometry. In other words, these two application cellular models must correspond to the same B-rep solid model, which represents the final part geometry. This geometric consistency is realized through mapping 2D or 3D application features to the corresponding cells in the shared unified cellular model.

- Two features may represent the same item at two abstraction levels, e.g. a central line or a cylindrical face of a hole. The consistency is maintained through specifying geometric or topological constraints on the related cells in the unified cellular model.

4.3.3 The Characteristics of the Unified Cellular Model

A unified cellular model *UCM* includes all geometries from different applications.

- It consists of a set of cells:

$UC_i : UCM = \left(\bigcup_{i=1}^{q} UC_i^0\right) \cup \left(\bigcup_{j=1}^{r} UC_j^1\right) \cup \left(\bigcup_{k=1}^{s} UC_k^2\right) \cup \left(\bigcup_{l=1}^{t} UC_l^3\right)$, in which UC^0, UC^1, UC^2, and UC^3 represent zero-dimension (0D), one-dimension (1D), two-dimension (2D), and three-dimension (3D) cells, respectively. Similarly, q, r, s, and t are the numbers of 0D, 1D, 2D, and 3D cells, respectively, in the unified cellular model.

- Each cell (except 0D cells) is bounded by a set of cells of a dimensionality lowered by one. On the other hand, a cell may exist independently without bounding any higher dimensional cell.
- The point sets of any two cells (of the same or different dimensionalities) do not overlap: $UC_i^a \cap UC_j^b = \phi$ $(0 \le a < b \le 3,$ or $(a = b) \wedge (i \ne j))$. In addition, a cell does not include its boundary, except for 0D cells.
- The cellular model obeys the Euler-Poincare formula for non-manifold geometric models [160]:

$$v - e + (f - r) - (V - V_h + V_c) = C - C_h + C_c$$

where v, e, f, r, V, V_h, V_c, C, C_h, and C_c are the numbers of vertices, edges, faces, rings, volumes, holes through volumes, cavities in volumes, cellular complexes, holes through cellular complexes, and cavities in cellular complexes, respectively.

Each application feature model uses the unified cellular model. The relations among these models are described as follows (see Figure 4.13, arrows in the figure represent "map" or "consist" relations):

- An application feature model *AFM* consists of a set of application features AF_i and other non-geometric entities NGE_j:

$$AFM = \bigcup_{i=1}^{m} AF_i \cup \bigcup_{j=1}^{n} NGE_j$$

  where *m* and *n* are the numbers of application features and non-geometric entities in this application feature model.
- An application cellular model *ACM* is created at runtime, which consists of a set of application cells AC_i: $ACM = \bigcup_{i=1}^{u} AC_i$, where *u* is the number of application cells in this application cellular model.
- Each application feature refers to a set of application cells. An application cell may belong to several application features, i.e. it records several features in its owning feature list. The geometries of an application feature correspond to 1D, 2D, or 3D cells.
- The point set of a cell comprises all points included in the cell. Then, the point set of an application cellular model is a subset of the point set of the unified cellular model. The cells of an application feature, after being inserted into the unified cellular model, could be further split by other applications. Hence, an application cell can be mapped to one or more cells in the unified cellular model. On one hand, for a particular application, one cell in the unified cellular model is mapped to at most one application cell. On the other hand, each cell in the unified cellular model is mapped to at least one application cell (and therefore at least one application feature). This mapping is realized through the owner attribute mechanism.
- The rule for determining cell nature applies to the unified cellular model, i.e. the nature of the latest feature in the owner list determines the nature of the cell.

As shown in Figure 4.13, all applications use this unified multi-dimensional non-manifold cellular model. The geometry of each application feature model is a particular aspect of the unified cellular model.

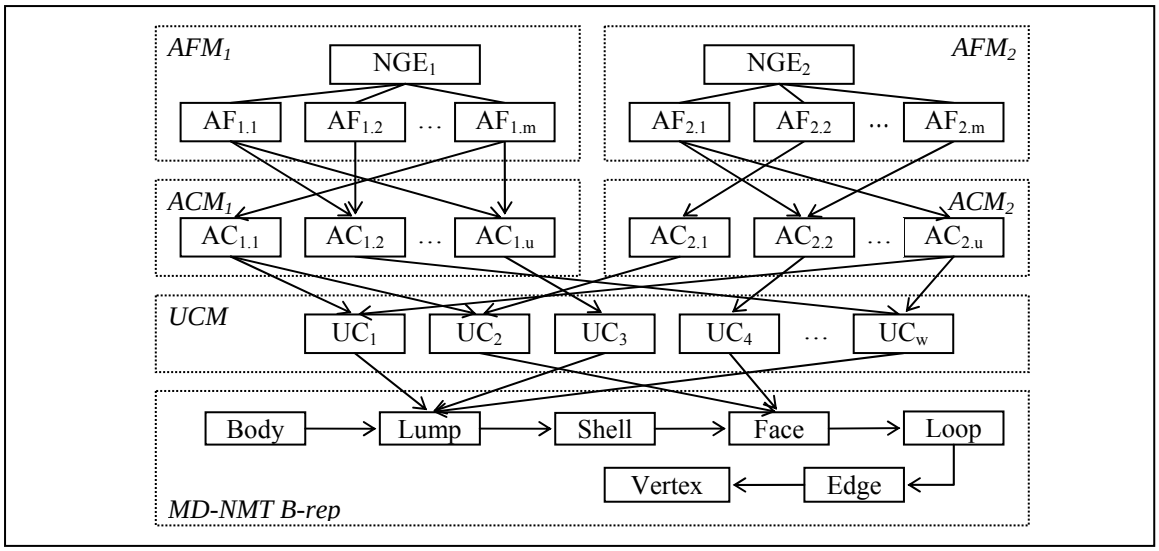


Figure 4.13 The hierarchal structure of the unified cellular model

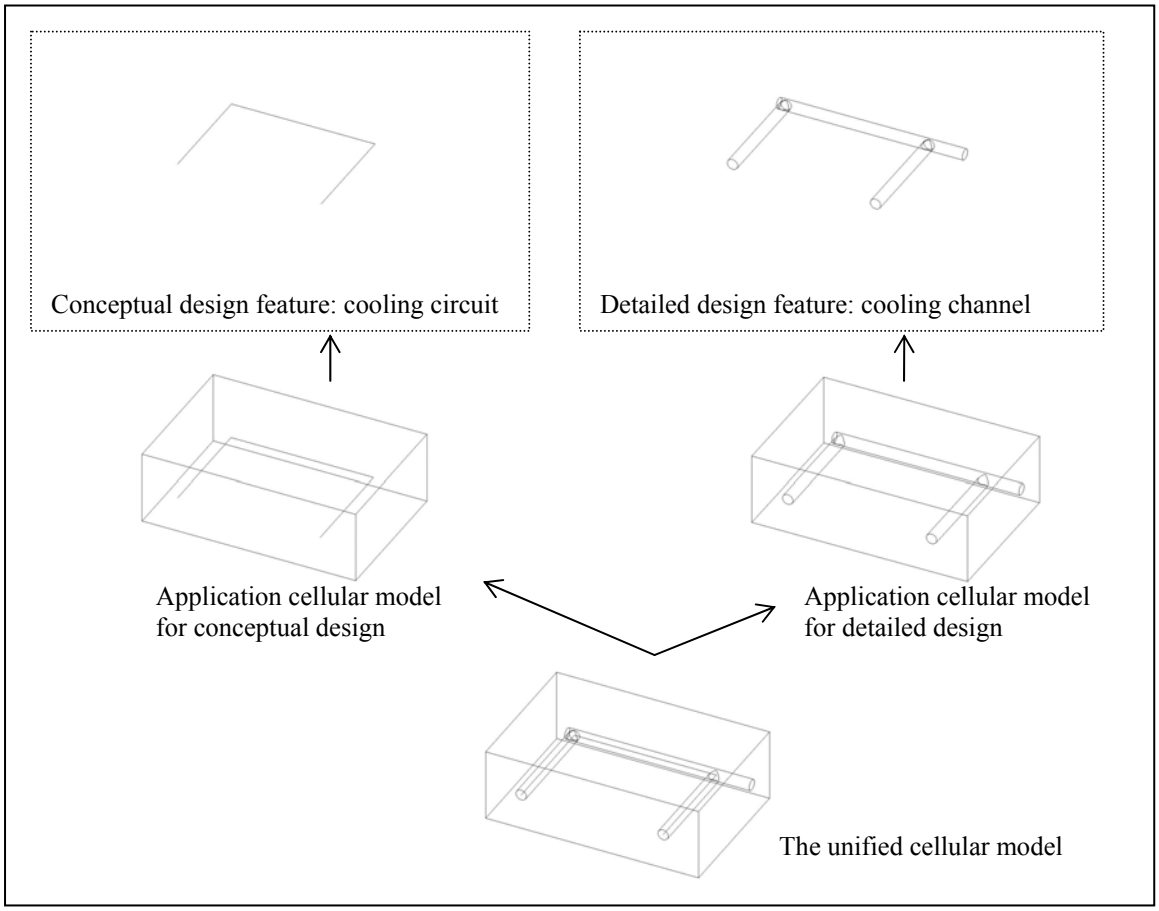


Figure 4.14 Link the conceptual and detailed designs using the unified cellular model

In Figure 4.14, the design of a cooling system of a plastic injection mould is used as an example to illustrate the idea. In the conceptual design stage, the cooling system is

represented as cooling circuits for cooling effect analysis while in the detailed design or process planning stage, the cooling system is in 3D for manufacturability analysis and process planning. The cooling circuits and the cooling channels are representations of the same cooling system in different abstraction levels. The geometries of these two feature models are kept consistent through the unified cellular model.

4.3.4 Two-Dimension Features and Their Characteristics

The idea in this sub-section is that unified cellular modeling scheme represents 1D, 2D, and 3D cells uniformly (they are referred to as edge, face, and solid cells respectively hereafter). Currently, the prototype system only handles face and solid cells (corresponding to 2D and 3D features respectively). 1D features are mentioned here but more research will be done in the future. Examples of 2D application features are:

- Conceptual design features, which represent functional areas in the product. The geometries of this kind of features are usually abstracted as pairs of interacting faces (Chapter 5). These faces correspond to partial faces in the detailed design.
- Assembly features, which represent grasping or mating areas of parts in assembly processes.
- Locating or clamping features, which represent locating or clamping faces during a machining operation.

Similar to solid cells, the major advantage of using face cells instead of geometric faces is that operations on face cells do not remove faces (or parts of faces) even they do not belong to the final boundary. For example, the non-regularized operations on faces and decomposition mechanism upon overlapping detection are available to face cells. A 2D feature is represented as a group of associated face cells with engineering semantics. For example, a locating feature is defined as a pair of faces associated with the constraints on accessibility, machining accuracy, non-interference, and minimizing setup changes. The geometry of a 2D feature is one or more surfaces. Two characteristics of 2D features are:

- A 2D feature has a nature attribute (additive or negative) that can be changed by feature interactions. A change of cell nature (from additive to negative or vice versa) requires the corresponding features to be validated. For example, a clamping feature represents a local area on a part that is used for clamping. When a clamping feature is

altered, its face cells may be split with the natures of some of the resulting face cells inverted. This may jeopardize the clamping feature's stability (sufficient area for clamping). Similar situations are encountered for functional, assembly, and locating features.

- Face cells corresponding to functional, assembly, locating, and clamping features have the same surface definitions as existing face cell(s). Hence, to simplify the implementation, it is assumed that newly inserted face cells and existing solid cells do not intersect. However, this is not valid for some CAE analysis applications, in which middle faces are commonly used.

When a 2D feature is generated, the corresponding face cell is also generated and inserted into the application and the unified cellular models. Figure 4.15 illustrates a simple example: the integration of a detailed design and a process planning model on the basis of a unified cellular model. The designed part is a block with a blind hole. The hole has a distance specification with face F2 as datum and a perpendicularity specification with face F1 as datum. The corresponding process planning model begins with a blank feature (larger than the part to allow machining). Other process planning features are:

- A surface-milling feature (due to the perpendicularity specification);
- A clamping feature (for surface-milling);
- A drilling feature and a boring feature (to meet the surface finish requirement of the hole);
- A locating feature and a supporting feature for the drilling and boring operations.

These 2D or 3D features are associated in the unified cellular model.

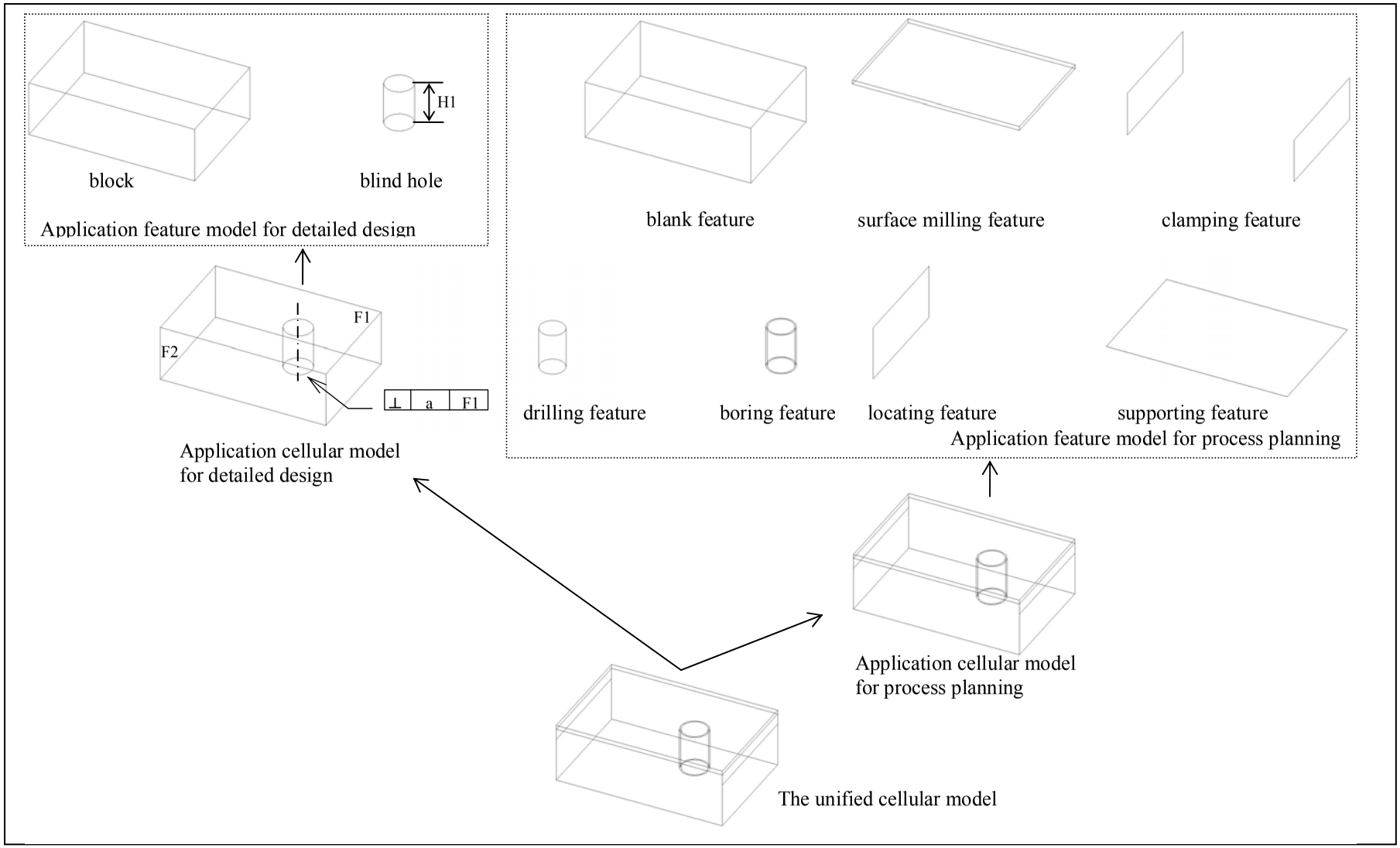


Figure 4.15 Integrating a detailed design and a process planning feature model using a unified cellular model

4.3.5 Relation Hierarchy in the Unified Cellular Model

Relations can be established on the cell level, the feature geometry level, and the feature semantic level respectively. Higher level relations are established on the basis of lower level relations. This relation hierarchy is as follows:

- The lowest level of relations is between two cells, which covers the following cases:
  - (i) The bounding relations among cells.
  - (ii) The bounding cells inherit the owner attributes of the bounded cell.
  - (iii) Two solid cells are adjacent if they are bounded by one or more common face cells. Two face cells are adjacent if they are bounded by one or more common edge cells.
  - (iv) Two adjacent edge or face cells may be part of the same curve or surface.
- Second level relations are topological relations between the geometries of application features. Note that a feature's dimensionality can be diverse depending on the application nature. Some topological relations between two application features are identified as follows:
  - (i) *Overlap*: After cellular splitting, two n-dimensional features are said to be *overlap* with each other if they use the same n-dimensional cell(s). A n- and a (n-1)-dimensional features are also said to be *overlap* with each other if they use the same (n-1)-dimensional cell(s).
  - (ii) *Adjacent*: Two different n-dimension features are defined as *adjacent* ones if they share (n-1)-dimensional cell(s) but do not overlap.
  - (iii) In a 3D feature, *adjoining area* refers to one or more faces (represented by the face cells), which are mathematically connected and defined on the same surface. For two 3D features A and B, feature A is said to be *completely adjacent* to feature B, if a feature A's adjoining area is fully enclosed by any of feature B's adjoining area. Figure 4.16 illustrates the complete adjacency of two 3D features. Other examples are:
    a. A single face in the detailed design corresponds to several functional faces in the conceptual design.
    b. A face in the process planning model corresponds to one or more faces in the detailed design.
    c. In plastic injection mould design, completely adjacent relations can be used to represent maps from the plastic part to core or cavity inserts as well as

electrode geometry. Such maps are commonly encountered in die casting, forging tooling, and fixture design as well.

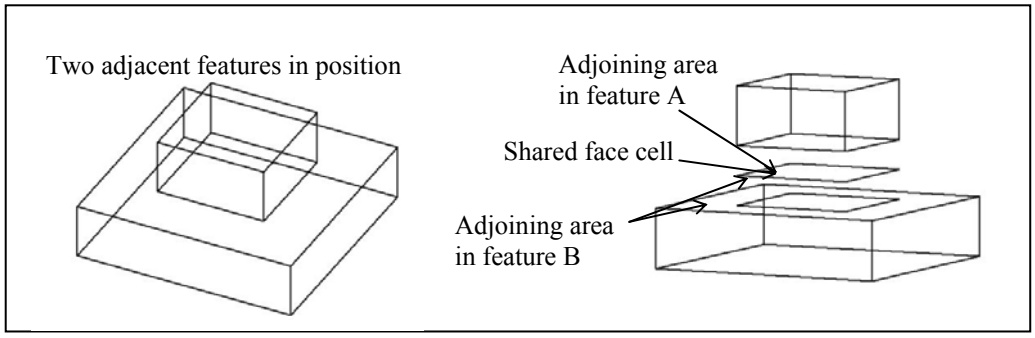


Figure 4.16 Completely adjacent relation between two 3D features

(iv) *Separated*: the geometries of two features do not spatially contact (overlap or are adjacent).

- Third level relations are semantic relations between application features. Relation types in this level are application specific. Examples are:

(i) *Splitting*. Figures 4.17(a) to (c) show a base block with a hole feature. The hole feature is further split by a vertical through slot feature. A similar situation for 2D features is shown in Figures 4.17(d) and (e), in which the original clamping feature is split by a newly inserted through slot feature. The middle face cell of the clamping feature becomes negative. The clamping feature must hence be checked for stability. This kind of relation between two interacting features is defined as a splitting relation [66]. Using the above-mentioned two lower levels of relations, the splitting relation can be described as:

a. The nature of the second feature is negative.
b. The two features overlap.
c. The insertion of the second feature splits the original single cell (additive or negative) of the first feature into several (at least three) cells, where the nature of at least one of the middle cell(s) is negative.

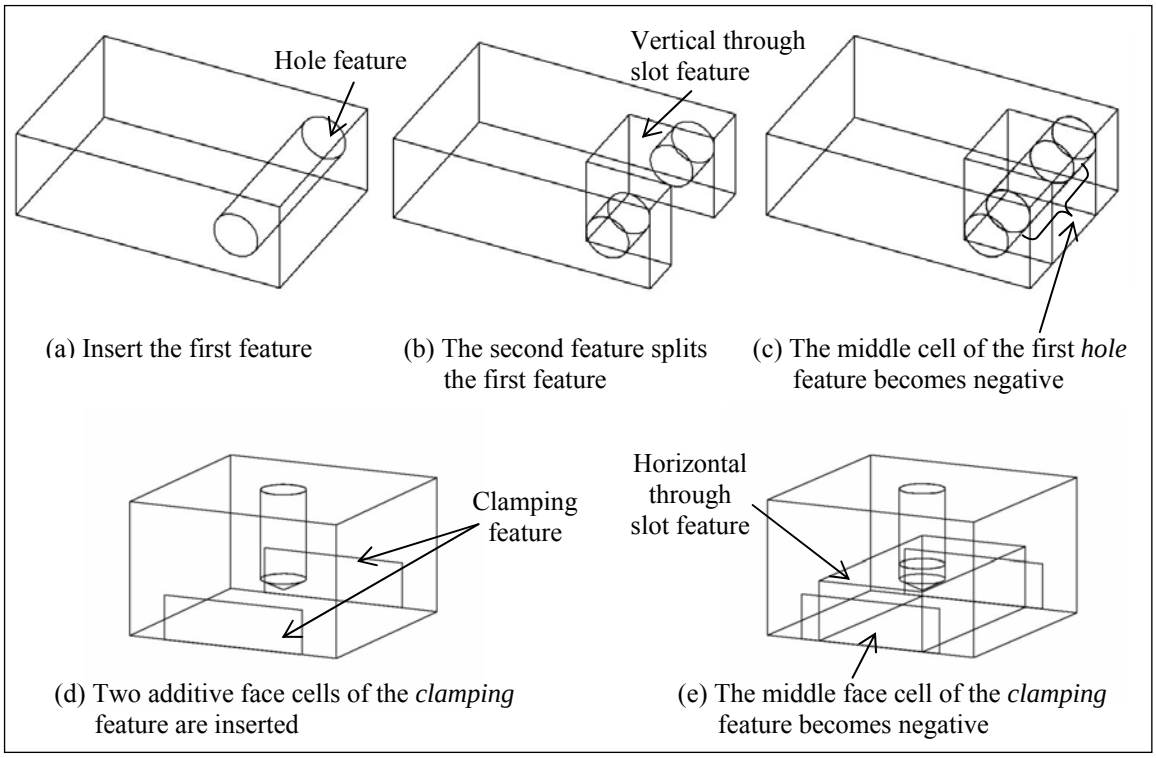


Figure 4.17 Splitting relation

(ii) Figure 4.18 (a) shows a base block with a blind hole feature and a vertical through slot feature. The relation between these two features is defined as a *transmutation relation* in [66]. Using the above-mentioned two lower levels of relations, the transmutation relation can be described as:

a. The nature of the two features is negative, but the nature of one of the bounding cells of the first feature, which represents the bottom face of a blind hole, is additive.
b. The two features overlap.
c. The insertion of the second feature splits the original single 3D negative cell into two 3D negative cells. The previous additive bounding 2D cell becomes negative.

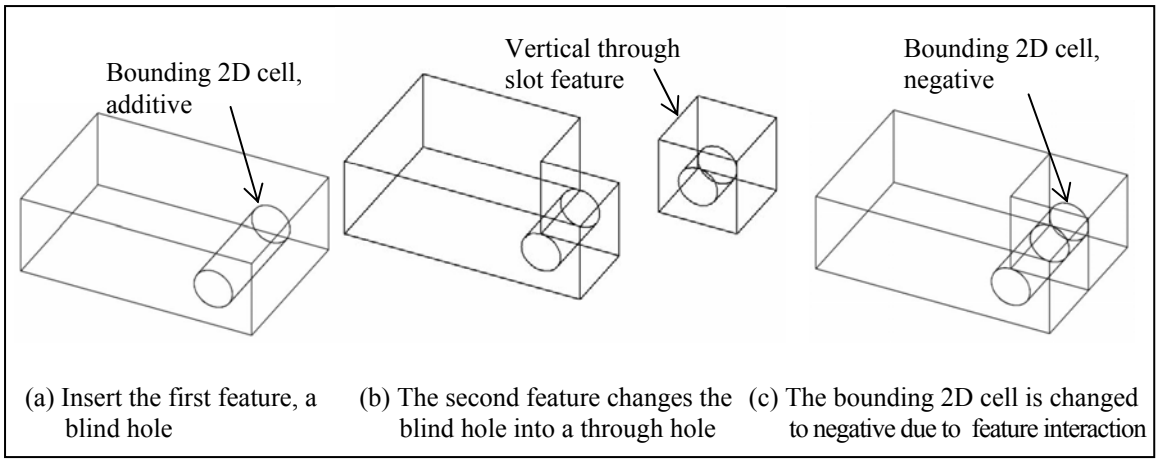


Figure 4.18 Transmutation relation

(iii) *Non-interference*. This relation specifies that two features cannot overlap with or are adjacent to each other. This constraint is satisfied if no cell in the unified cellular model has both of these two features in its owner list. This constraint is commonly used in product design or manufacturing activities. For example, a process planning feature cannot interfere with the corresponding clamping features.

4.4 Summary

The basic idea of the proposed unified feature-based product modeling scheme is using data association and data sharing to maintain the validity and consistency of product models. Linking engineering knowledge to product geometry and integrating multiple applications are two important characteristics of the proposed scheme. This chapter describes these two characteristics:

- A rule-based expert system is used to manipulate engineering knowledge. The relations between engineering knowledge and features are explored.
- A unified cellular model is used to share geometric data among applications.

With the unified feature definition and the rule-based expert system, the conceptual design features and process planning features are defined in the next chapter.

Chapter 5
Application Feature Definition

All application features are subclasses of the unified feature. At the same time, they also have application specific attributes, methods, constraints, and semantics. Two of these features are defined in this chapter: conceptual design features and process planning features. In addition, relations among application specific entities are described. The purpose is to use these inherent relations to establish a dependency network to facilitate change propagation (Chapter 6).

5.1 Conceptual Design Feature

The conceptual design stage has different characteristics than the detailed design stage. The traditional design features need to be extended to support conceptual design activities.

5.1.1 Why Incorporates the Earlier Design Stages

The design process of a mechanical product is generally divided into three stages: conceptual design, embodiment design, and detailed design (Pahl & Beitz [193]).

- Conceptual design results in the *specification of principles* through abstracting the essential customer requirements, establishing function structures, searching for suitable working principles, and then combining those principles into a working structure.
- Embodiment design results in the *specification of structures* of a technical system in line with a technical and economical analysis.
- Detailed design results in the *specification of assembly, sub-assembly and parts*, which includes the arrangement, shapes, dimensions, and materials of all individual parts as well as the assessment of the production possibilities and cost estimation.

Pahl and Beitz also pointed out that “it is possible that there are several principle solution variants” in the conceptual design stage and several preliminary structures in the embodiment design stage [193]. The designer usually selects the most promising variant at the moment to develop (Figure 5.1). However, the exploration of other design solutions must also be supported in a systematic way [150].

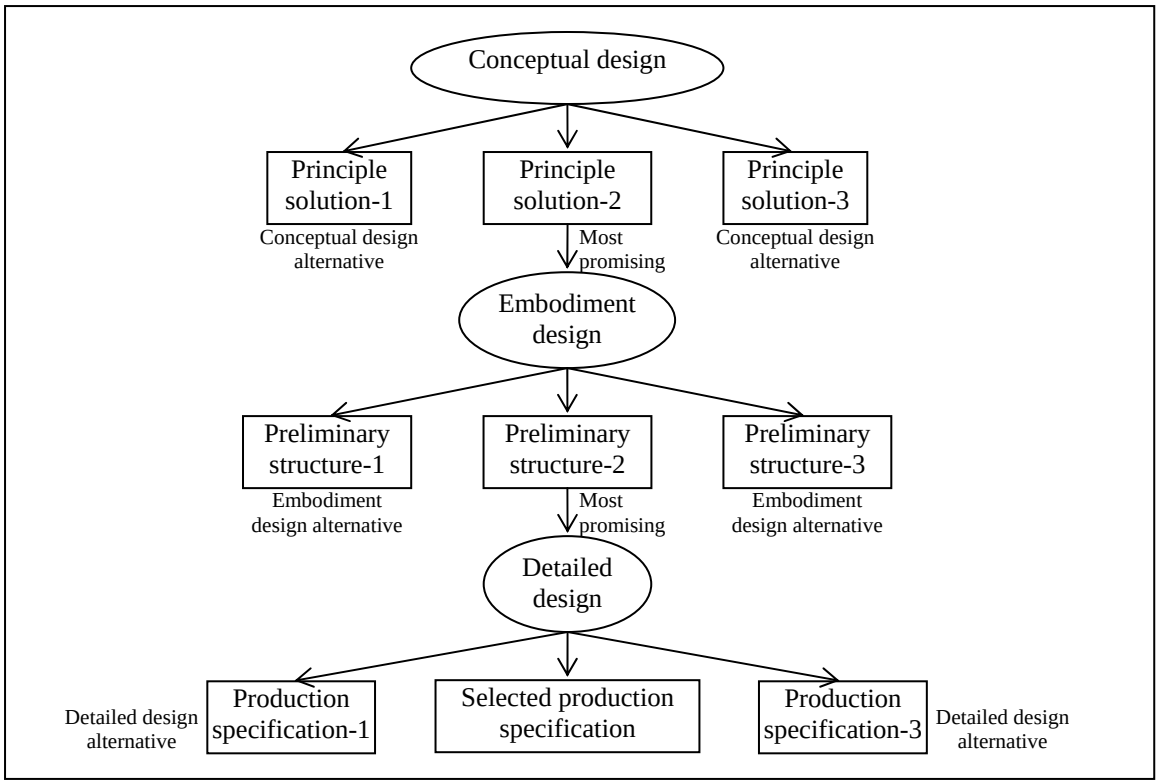


Figure 5.1 Design stages and design alternatives

Traditional CAD systems do not support the conceptual and embodiment design stages – only the detailed geometric design. However, design intent – emerging in the earlier stages – must be recorded appropriately because they are important for exploring other design solutions, design modifications, and design optimization. It is inherently difficult for traditional CAD systems to record design intent because their data types and data structures were developed for geometry manipulation. Parametric and constraint-based CAD systems permit the designer to partially record their intent (only for the most promising embodiment design solution) as detailed design parameters, geometric or algebraic constraints. From the perspective of recording design intent, traditional CAD systems have two shortcomings:

- The alternative principle solutions and other preliminary structures are not recorded in the design.
- Even for the most promising preliminary structure, the respective design intent is lost if it cannot be fully translated into design parameters or constraints.

These shortcomings result in an incomplete embedment of design intent. To support design optimization and modification, more comprehensive design intent representation,

i.e. a record of the causal relations between the design knowledge (functional design or DFX rules) and the product specifications, is necessary. An association of earlier design stages (conceptual and embodiment designs) with the detailed design stage based on a unified feature-based product modeling scheme provides a solution to these problems. The following characteristics distinguish the models of these earlier life cycle stages from traditional detailed designs:

- These models focus more on determining the high-level abstract product specifications (working principles, structures, layout, parameters, and critical specifications) than the detailed and complete product specifications (detailed geometries and dimensions).
- These models use symbolic reasoning more extensively than numeric calculations.
- 1D, 2D, and 3D geometries coexist in them.

Previous research on conceptual design usually does not address the relations between functionality and detailed geometry. The unified feature-based product modeling scheme includes a definition of conceptual design feature as well as an association mechanism between the conceptual design and the detailed design feature models.

5.1.2 Workflow during the Conceptual Design Stage

Major tasks in the conceptual design stage include function decomposition, functions to physical structures mapping, as well as the determination of critical geometries and specifications. These tasks are often carried out with the assistance of conceptual design knowledge base. To provide a basis for the definitions of conceptual design features, major entities, activities, and workflow processes during the conceptual design stage are analyzed as follows.

Function Decomposition

The conceptual design of a mechanical product begins with the specification of the required *functions* and ends with a working structure. In this research, a product function is defined as “what a product is supposed to achieve” while a working structure is a group of combinations of the physical effect with the geometric and material characteristics [193]. Initial overall function specifications need to be decomposed into

sub-functions until the designer can match them to known design solutions. During this decomposition process, necessary supplementary or auxiliary sub-functions are identified and generated as well (Zhang et al. [194], Deng et al. [195]). The compatible combination of these sub-functions produces a function structure. There are two major types of relations in a function structure:

- *Decomposition*, i.e. a function is decomposed into several sub-functions. These sub-functions may have AND (coexisting) or OR (alternative) relations. For the OR relations, the most promising variant is selected for further development.
- *Sequential*, i.e. some sub-functions can be and have to be evaluated before others.

The result of function decomposition is a set of primitive functions, which are not further decomposed. For mechanical products, a primitive function is usually realized by a particular set of interacting geometries. Especially, the most commonly used type of geometries in the conceptual design stage is the "face". Hence, the term "interacting faces" instead of "interacting geometries" is used hereafter to refer to the geometries of conceptual design features.

Interacting Faces, States, and Behaviors

To realize a primitive function, the interacting faces must have appropriate properties and inter-relations. The properties may be unchangeable (shape, dimensions, material, etc.) or changeable (position, orientation, etc.). The interactions are based on physical laws. Due to the difficulty of directly linking product functions to product geometries, some researchers suggested to use the behaviors of interacting components as a medium. The product *behavior*s are generally defined as "how a product achieves the required functions". Behaviors are described by the *states* of the interacting faces, where state is a set of changeable properties of a particular face. Behaviors can be static or dynamic.

- *Static behaviors* describe static states of a face. For example, if a particular position of a face is necessary to realize a required function, then being kept in this particular position is a kind of required static behavior of this face.
- *Dynamic behaviors* of a face describe a particular set of state transitions of the face. For example, a face moves from one position to another position.

Interactions between the corresponding faces can result in state transitions and hence part behaviors. Behaviors are objective. In other words, for two interacting faces with specified properties (changeable and unchangeable) and inter-relations, a particular

behavior is displayed under the particular external inputs according to physical laws. Note that the side effects, i.e. unwanted behaviors, are not considered here.

Using the behaviors as medium, a primitive function is described using the following elements:

- Unchangeable properties of the interacting faces;
- Changeable properties of the interacting faces as well as their initial values;
- Inter-relations between the interacting faces;
- Behaviors, which are unchanged states or a sequential of state transitions;
- Underlying physical laws;
- External inputs.

Working Structures in Conceptual Design vs. Assemblies in Detailed Design

Interacting faces represent the functional areas of a product. Usually the detailed geometric characteristics of these interacting faces are not determined at the conceptual design stage. However, the critical properties (in the view of the required functions), such as the face type, relative position and orientation, must be specified at this stage. The working structure comprises these roughly-defined interacting faces. A working structure is concretized and transformed into a system (an assembly for a mechanical product with sub-assemblies and components) in the detailed design stage. For example, as shown in Figure 5.2, the interacting faces are scattered over several components. At the same time, function clustering may cause that certain detailed design features accommodate more than one conceptual design features. Interacting faces must have their corresponding detailed design faces. However, a detailed design component may be created for the purpose of realizing auxiliary functions, which are unspecified in the conceptual design stage. The corresponding relations between the conceptual design faces and the detailed design faces are injective but not surjective.

The above analysis shows that geometric model alone can not represent design intent completely. For example, in Figure 5.2, components C1.2, C1.3, and C2.1 must exist simultaneously with the specified inter-relations as they realize the primitive function F3.1 together. Current CAD systems cannot record such *co-existence* relations. Furthermore, the realization of a function depends on a particular set of behaviors, which consequently depend on product geometry, their inter-relations, as well as physical laws.

Because these *causal* relations are more suitable to be represented as function and working structures, it is hence difficult to represent them without referring to the conceptual design and its features.

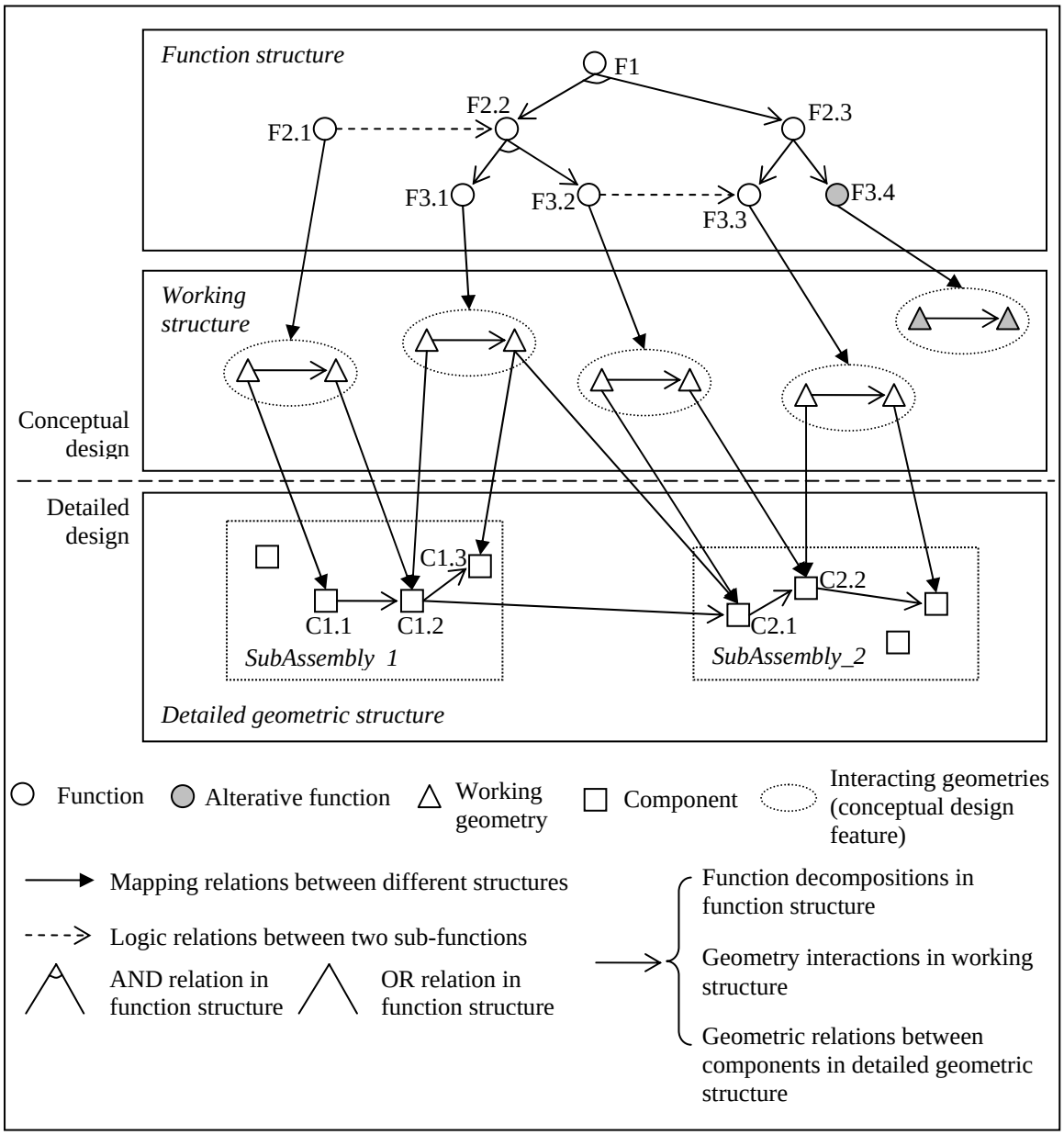


Figure 5.2 Mapping from function specifications to detailed geometries

5.1.3 Definition of the Conceptual Design Features

The relations between the primitive functions, the required behaviors, and the properties of the interacting faces determine the critical characteristics of the product

(Figure 5.3). They are modeled as *conceptual design features*. Conceptual design feature model provides a framework for detailed design.

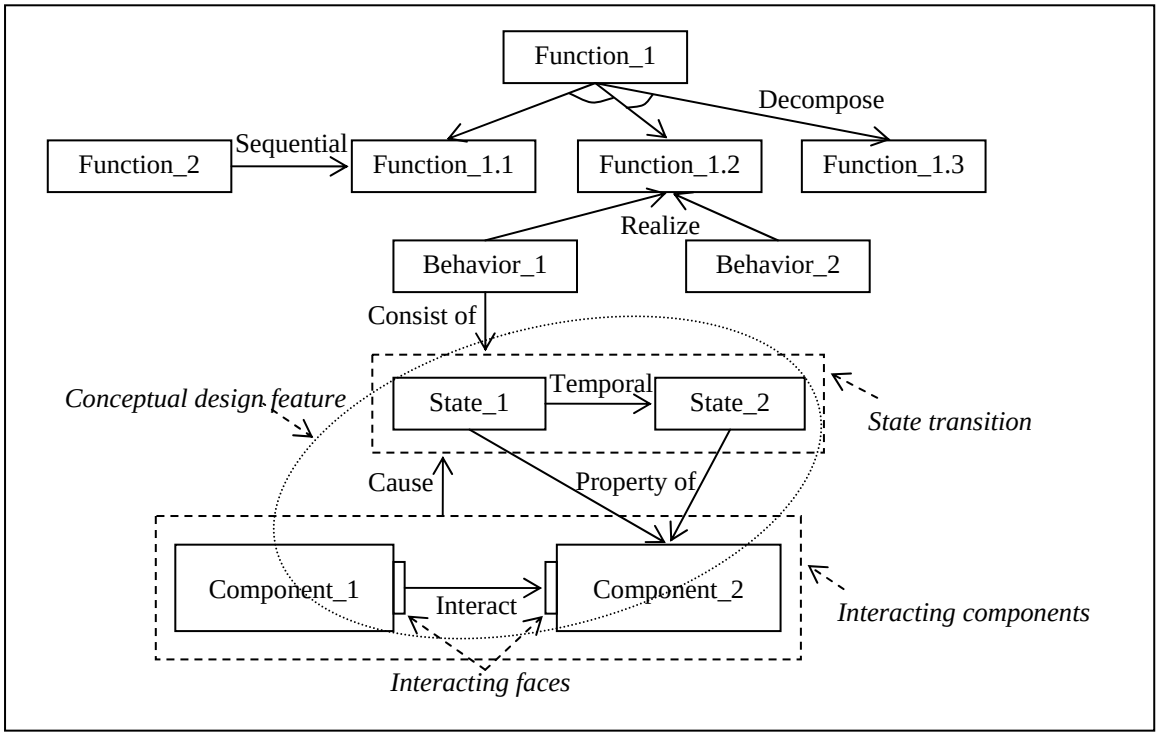


Figure 5.3 Conceptual design process and conceptual design feature

Conceptual design features are defined from the following perspectives:

- Each conceptual design feature corresponds to a particular primitive function.
- Each conceptual design feature includes a set of interacting faces.
- Critical properties of and the inter-relations between the interacting faces are specified.
- Properties may possess default values or value ranges.
- Required behaviors are represented as the initial and final states of the interacting faces.

Being sub-classes of the unified feature class (Chapter 3), conceptual design features inherit the generic fields and methods. As the product's physical structure usually lacks details in this stage, conceptual design features have to allow the representation of incomplete, inaccurate product specifications (such as the value ranges used in ([104], [154]). This includes parameterized or non-manifold geometries. In such cases, default

values, interpolations, or even symbolic objects may be used to create and represent conceptual design features. The conceptual design feature model gives an abstract impression of the product.

The unified feature-based product modeling scheme uses an embedded rule-based expert system to deal with the issues like function decomposition, function to physical structures mapping, and determination of critical product specifications. A cellular model is used to represent the abstract geometries referred by the conceptual design features. The dependency relations between the primitive functions and the conceptual design features are used to justify the existences or modifications of the conceptual design features. The detailed design reflects the required product functions through its relations with the conceptual design.

5.1.4 Rule-Based Constraints in Conceptual and Detailed Design Stages

Rule-Based Constraints in the Conceptual Design Stage

The input to the conceptual design stage is the product functionality. The output is a set of inter-related conceptual design features as well as the corresponding geometries (if they are specified). Rule-based constraints (see Chapter 4) are used in function decomposition, function feasibility evaluation, function-to-feature mapping, and critical feature property generation:

- Function decomposition rules decompose a function F to a set of sub-functions SF: $\{F_1\} \rightarrow \{SF_1, SF_2, ..., SF_n\}$. This decomposition process is carried out level by level iteratively. At each level, primitive functions are identified. For non-primitive functions, the decomposition is continued until all the sub-functions are primitive. For example,

  Rule_1:

  IF (FU_1 exists) and (FU_1.*primitive* is false)

  THEN (generate FU_2, FU_3,…, FU_n as sub-functions of FU_1)

  where FU represent design functions and FU_1.*primitive* is a member variable of the function class, which indicates whether FU_1 is a primitive function (true) or not (false).

When the rule fires, two rule-based constraints are generated:

(i) The sub-functions depend on the parent function.

(ii) All sub-functions must co-exist to fulfill the parent function.

- Feasibility evaluation rules check whether the preconditions (usually provided by other functions) of a function are satisfied. For example,

  Rule_2:

  IF ($FU_1.pc$ does not exist)

  THEN ($FU_1.feasibility$ is false)

  Rule_3:

  IF (FU_1. *feasibility* is false) and ($FU_1.pc$ matches the output of FU_2 in the function library)

  THEN (generate FU_2) and ($FU_1.feasibility$ is true)

  where $FU_1.pc$ represents the preconditions (*pc*) of function FU_1 while $FU_1.feasibility$ is a member variable of the function class, which indicates the feasibility of a function FU_1.

  Firing rule_3 generates a rule-based constraint between the supplementary function FU_2 and the main function FU_1.

- Function-to-feature mapping rules create the respective conceptual design features if a feasible primitive function is generated. For example,

  Rule_4:

  IF ($FU_1.primitive$ is true) and ($FU_1.feasibility$ is true)

  THEN (generate *FE*)

  where *FE* is the conceptual design feature set corresponding to FU_1.

  Firing this rule generates a rule-based constraint, i.e. conceptual design features depend on the corresponding primitive functions.

- Critical property rules decide the values or the value ranges of conceptual design feature properties. These specifications include properties of the interacting faces as well as relations between the interacting faces [172]. For example,

  Rule_5:

  IF (FU_1 is “transmit force”)

  THEN (generate *CONS1* (two interacting faces have the same surface type)) and (generate *CONS2* (two interacting faces have sufficient contact area))

Rule_6:

IF (FU_1 is "transmit torque") and (*FE1.IFACE.surfaceType* is "cylindrical")
THEN (generate *CONS1* (two interacting faces have interference fit condition)) and (generate *CONS2* (the cylindricity form tolerance of the interacting faces is higher than a specified value))

where *CONS1* and *CONS2* are numerical constraints, *FE1.IFACE* represents the interacting faces of this conceptual design feature while *surfaceType* is one of the properties of the interacting faces.

Firing these rules also generates rule-based constraints; hence the critical feature properties depend on the corresponding primitive functions. Besides constraints, firing a rule may also decide other feature properties, such as surface finishes or material types.

The associations (established on the basis of rule-based constraints) between the rules, features, and functions are illustrated in Figure 5.4.

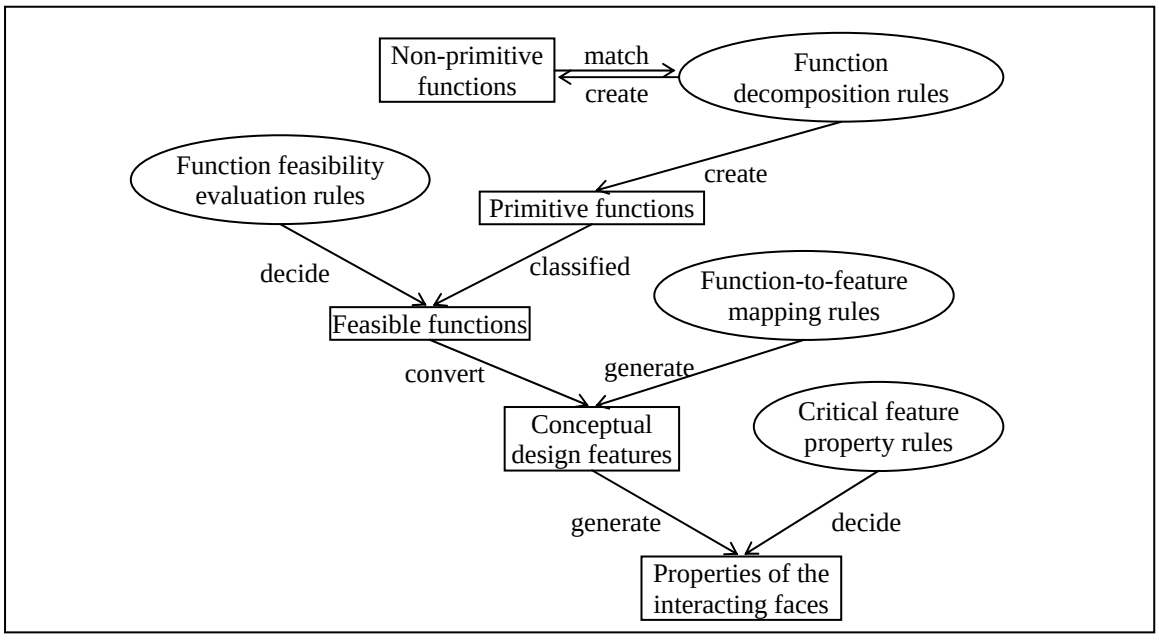


Figure 5.4 Rule-based constraints in the conceptual design stage

Rule-Based Constraints in the Detailed Design Stage

The input to this stage is the conceptual design feature model. The output is the detailed design feature model, which is suitable for manufacturing. Rule-based constraints are applied in two kinds of situations:

- Most detailed design features are created to materialize or complete the conceptual design (Figure 5.5). Besides fulfilling design functions, a product design needs also to consider the requirements of downstream stages (DFX functions). In other words, some features are generated to fulfill DFX functions. Hence, detailed design features depend on the corresponding conceptual design features (representing design functions), or DFX functions as well as the function fulfillment rules. These dependency relations are represented as rule-based constraints.

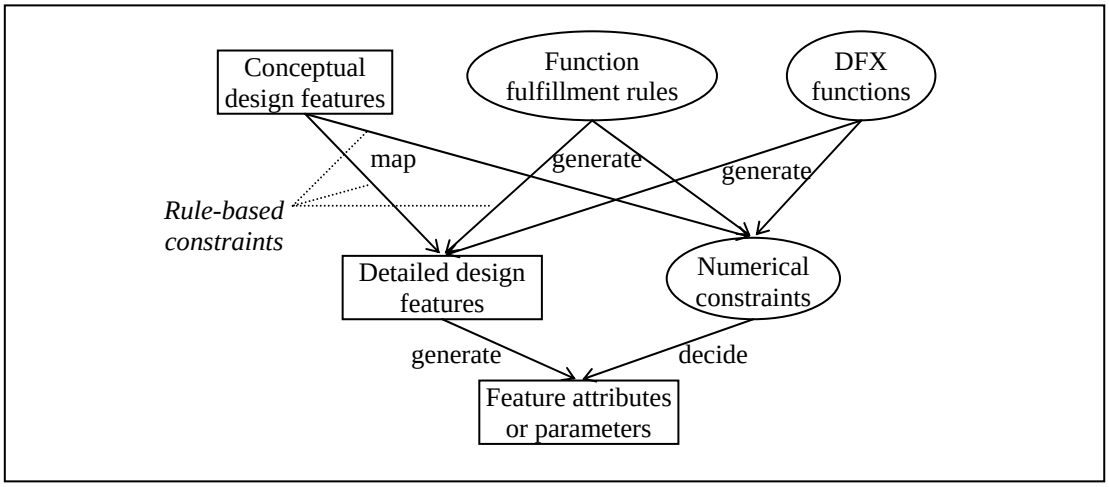


Figure 5.5 Constraint-based associations in the detailed design stage

- If numerical constraints are generated from DFX functions using function fulfillment rules, the presence of the numerical constraint depends on the presence of the corresponding DFX functions and the function fulfillment rules (Figure 5.5). The dependencies are more appropriately represented as rule-based constraints. The values of attributes and parameters of each detailed design feature are determined by solving the numerical constraints.

5.2 Process Planning Features

The feature concept originates from integrating design and process planning applications. However, the definitions and usage of the traditional machining features do not support process planning very well:

- Traditional machining features are volumes (holes or slots, etc.), which correspond to specific machining methods (drilling or milling, respectively). Usually, machining features are generated before processes are planned and are not a result of it.

Machining features do not reflect the detailed procedures of process planning. Hence, it is difficult to use machining features to represent process planners' strategies.

- The shapes of machining features are predefined. These predefined shapes are matched against the design in order to recognize machining features. However, in practice, the shapes produced by machining operations are based on the structural context (shape and topology) of the part as well as the selected machining tools. For example, an end-milling operation can produce a pocket of arbitrary profiles as long as the machining constraints (e.g. corner radius) are satisfied. Hence, the shapes of machining features must be able to be defined flexibly.

Process planning features are proposed here to represent the procedures and results of process planning activities. The major difference between the proposed process planning features and traditional geometric machining features is that process planning features are the results of process planning while the machining features are the start point of process planning. Hence, the process planning feature model can represent a process planner's strategy more completely through associations to machine tools, cutters, setups, machining sequence as well as process planning rules.

This section concentrates on illustrating that process planning features are derived from and are linked to the process planning procedures. Revealing the inherent dependency relations between data (machine tool, cutter, process planning feature, process planning rules, etc.) is the focus of this section. Hence, the following process planning algorithm generates only likely feasible and not necessarily optimal process plans. This algorithm only serves the purpose to identify process planning features and to support the case study (the algorithm could be revised in the future without major changes to the process planning feature definitions). In addition, to simplify the problem, it is assumed that the following process planning algorithm is for prismatic parts with 2.5-dimentional design features, a 3-axis machining centre, and standard cutters. A standard vice is used as fixturing device.

5.2.1 Definitions of Major Entities in a Process Plan

Entities of a process plan are defined as follows (Wang & Li [196]):

- A *machining operation* is characterized by unchanged machine tool and unchanged workpiece.
- A *setup* is part of a machining operation performed without re-locating and re-clamping the workpiece.
- An *operation element* is a component of a setup, which is performed without changing the cutter, the machined workpiece face, cutting speed, and feed rate. Note that the machined workpiece face may include a set of faces that are machined continuously using the same cutter.
- If the thickness of the removed layer of the workpiece exceeds the permitted depth of cut of a machine tool, an operation element is divided into several cuts. A *machining cut* is a single and complete cutting motion of the cutter in the direction of the feed along the machined face.

The process planning features defined here are suitable for material removal cutting operations only, such as milling and drilling operations. A process planning feature is the basic unit of a process plan and corresponds to a machining cut (Figure 5.6).

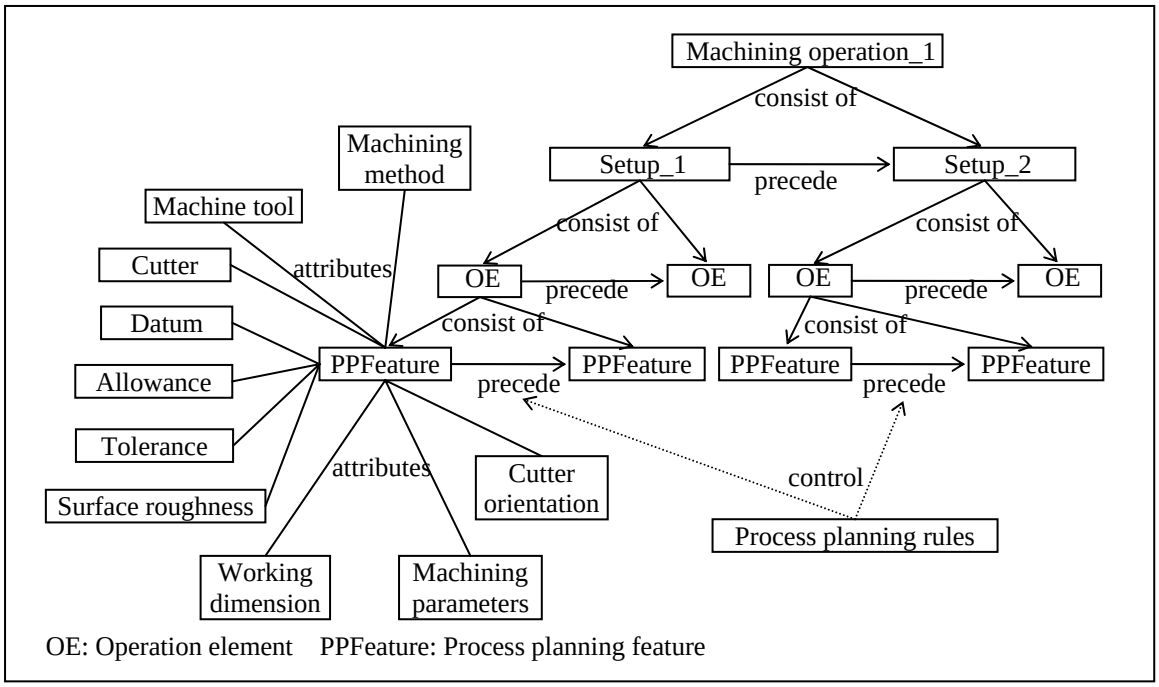


Figure 5.6 Semantic net of the process planning features

The geometry of a process planning feature is defined as the volume removed from the workpiece in the machining cut. Together with the fired process planning rules and

other process planning specific entities, process planning features can collectively represent process planners' strategy. Being sub-classes of the unified feature, process planning features inherit the generic fields and methods.

5.2.2 Workflow during the Process Planning Stage

Major steps of process planning include:

- Selecting the blank and its manufacturing method.
- Choosing machining methods and machining routes for each part face to be machined.
- Selecting the machine tool, the cutter, and the cutter orientation for each machined face.
- Determining operation elements by grouping machined faces, which can be machined continuously using the same machine tool, cutter, and cutter orientation.
- Determining workpiece locating and clamping methods for each operation element.
- Arranging the machining sequence of operation elements.
- Generating setups by grouping operation elements, which can be machined using the same machine tool, cutter orientation, the workpiece locating and clamping methods.
- Arranging the sequence of setups.
- Determining machining parameters for machining cuts.
- Generating tool paths and NC codes.

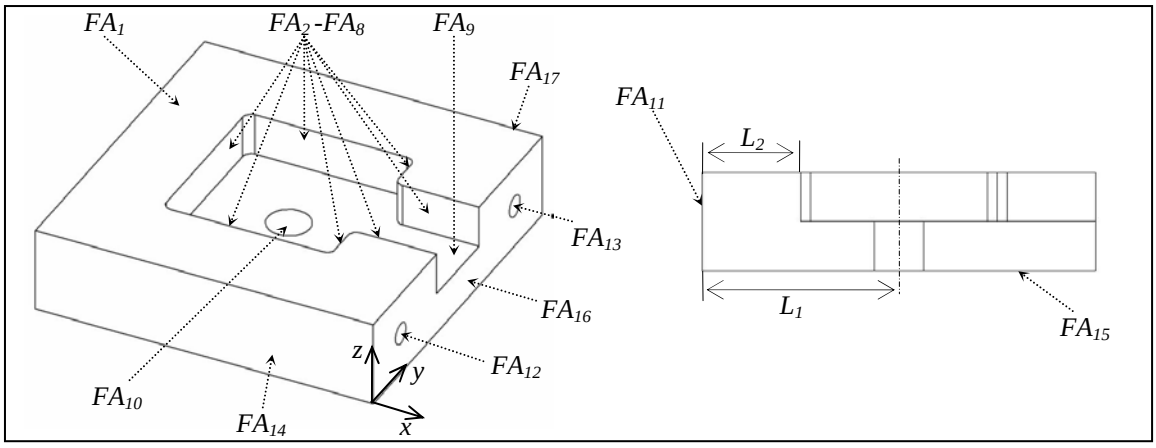


Figure 5.7 The detailed design of the example part

Some of these steps are possibly reiterated and each step usually produces several alternatives. For example, in step 2, a particular face usually can be produced using

different combinations of machining methods. These machining methods further correspond to several possible combinations of machine tools and cutters. These choices are evaluated and selected along with the addition of machining constraints in order to find a good solution. A simple part (shown in Figure 5.7) is used to illustrate the procedures. Note that the coordinate system shown in the figure is local to the part.

Choosing Machining Methods and Machining Routes for Each Machined Face

Part faces that need to be machined as well as the blank (raw material) are determined first. For the example part, if a rectangular bar is chosen as the blank, faces 1 to 15 (Figure 5.7) need to be machined (assuming the faces on the blank corresponding to face 16 and 17 already meet the design specifications). The surface type (*PL* for planar surface and *IC* for internal cylindrical surface) and the normal of each face (FN_{FA}, extracted from the design) are listed in Table 5.1.

Table 5.1 Surface type and face normal of each machined face

	FA_1	FA_2	FA_3	FA_4	FA_5	FA_6	FA_7	FA_8	FA_9	FA_{10}	FA_{11}	$FA_{12,13}$	FA_{14}	FA_{15}
ST_{FA}	*PL*	*PL*	*PL*	*PL*	*PL*	*PL*	*PL*	*PL*	*PL*	*IC*	*PL*	*IC*	*PL*	*PL*
FN_{FA}*	*+z*	*+x*	*-y*	*-x*	*-y*	*+y*	*-x*	*+y*	*+z*	*+z/-z*	*-x*	*+x*	*-y*	*-z*
* *The face normal is local to the part coordinate system.*														

As shown in Figure 5.8, a particular surface type ST_{FA} corresponds to a particular set of (including alternative) machining methods $MM_{FA} = \{m_1, m_2, ..., m_n\}$. For example, a planar face can be machined using milling, broaching, or grinding while an internal cylindrical face can be drilled, bored, or ground.

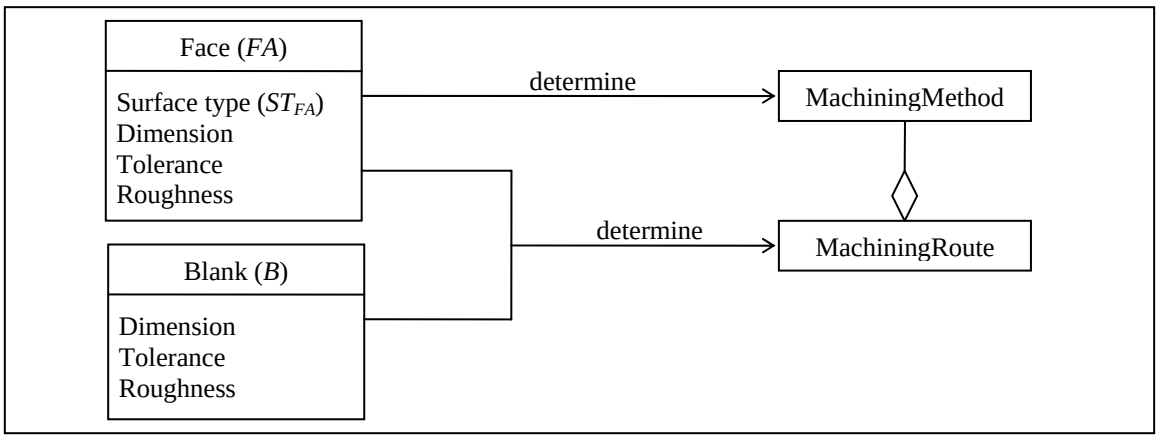


Figure 5.8 The machining methods and machining routes for each machined face

The dimension DIM_{FA}, tolerance TOL_{FA}, and surface roughness SR_{FA} specifications of a part face as well as those (DIM_B, TOL_B, and SR_B) of the respective face (if applicable) on the blank further determine the possible machining routes MR_{FAi}. A machining route is an ordered set of machining methods for a single machined face to meet its design specifications (Figure 5.8), e.g. drilling before boring or rough milling before semi-finish milling. Usually, there are many possible machining routes for each machined face.

Selecting the Machine Tool, Cutter, and Cutter Orientations for Each Machined Face

The selected machining method *m* determines a set of suitable machine tools *MT*. For example, a milling operation can be performed on a horizontal (vertical) milling machine or a horizontal (vertical) machining center (Zhang et al. [197]). The required cutting force and the size of the workpiece further restrict the selection. A set of alternative cutters *T* used to produce a machined face are determined by the selected machining methods and machine tools (Figure 5.9). For each selectable cutter $T_i \in T$, the spatial relations *FFR* between this face and other adjacent faces restrict the usable cutter sections *TS*, such as side, end, or corner sections. Furthermore, together with the normal of the face FN_{FA}, each cutter section (if applicable) determines the cutter orientation *TO*. The cutter orientation is the axial direction of the cutter during machining. Note that at this stage cutter orientations are specified using the part's local coordinate system.

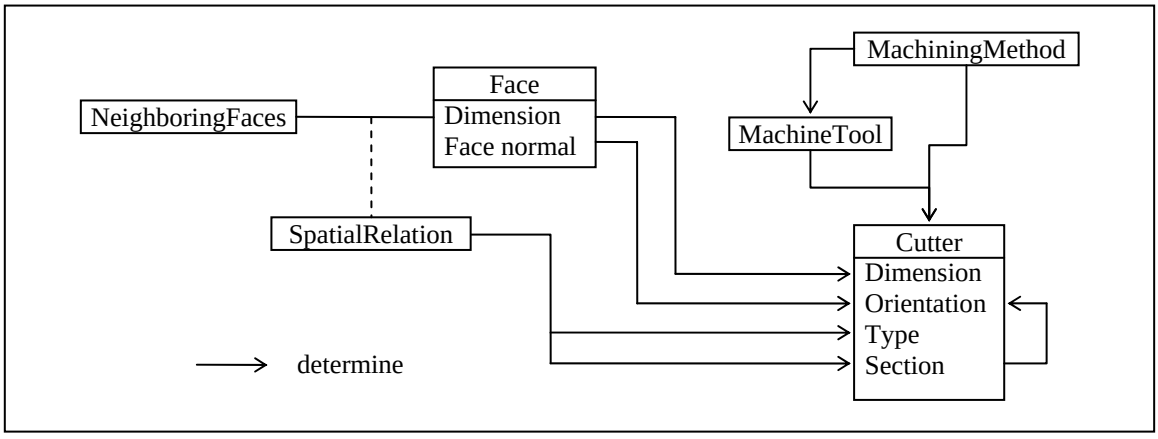
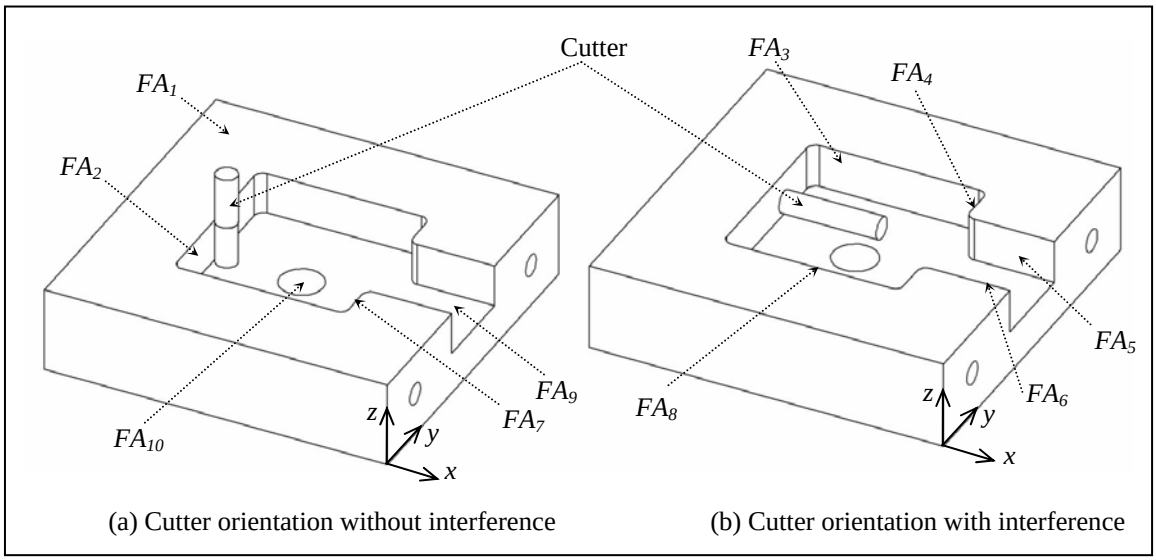


Figure 5.9 Determine machine tool, cutter, and cutter orientation

For instance, in the example part, the initial cutter set for face FA_2 include end, face, angle, and side mill. However, due to the spatial relations between FA_2 and its

neighboring faces (FA_1, FA_3, FA_8, and FA_9), only end mill and side mill can be used (Figure 5.10a); angle and face mill are not possible due to interference. In addition, only the side section of the end mill or side mill can be used to machine FA_2 (Figure 5.10b).



(a) Cutter orientation without interference (b) Cutter orientation with interference

Figure 5.10 Constraints on the cutter

After the machine methods are selected, possible relative orientations between the face and a selected cutter are determined. The possible combinations of cutters, cutter sections, and cutter orientations for each machined face are listed in Table 5.2.

Table 5.2 Cutter/cutter section/cutter orientation combinations for each machined face

FA_1	$FA_{2,3,4,7,8}$	$FA_{5,6}$	FA_9	FA_{10}	FA_{11}	$FA_{12,13}$	FA_{14}	FA_{15}
EM/end/+z SM/side/xy*	*SM(EM) /side/+z*	*SM/side/+z EM/side/+z SC/end/+-y*	*EM/ end/ +z*	*D(BT) /+z,-z*	*EM/end/-x SM/side/yz*	*D/+x*	*EM/end/-y SM/side/xz*	*EM/end/-z SM/side/xy*
EM – end mill, SM – side mill, SC – slot cutter, D – drill, BT – boring tool *end – end section of a tool, side – side section of a tool* *+z – the cutter orientation is along with the positive z direction* *xy – the cutter orientation is parallel to the xy plane.* ** The cutter orientation is local to the part coordinate system.*								

Determination of Operation Elements by Grouping Machined Faces

It is possible that a group of neighboring faces (with respect to the available cutters) can be machined using the same machine tool, cutter, cutter orientation, locating and clamping methods. These faces are grouped into a single operation element *OE*. For each machined face, an initial operation element is generated. Its neighboring faces are analyzed one by one. If any of its neighboring faces can be produced using the same

machine tool, cutter, cutter orientation, locating and clamping methods, the face is added to the initial operation element. The generated operation elements may overlap. The overlapping portions are removed from all but the longest operation element [134]. This heuristic is for the purpose of reducing the number of operation elements only. It may not be optimal from other perspectives. The generated operation elements for the example part are listed in Table 5.3. This face-by-face analysis approach is used to overcome the limitation of the traditional approaches that require the predefined feature shapes.

Table 5.3 Operation elements

Operation elements	*Faces to be machined*	*Cutters*	*Cutter sections*	*Cutter orientations (part coordinate system)*
OE_1	FA_1	*End mill (side mill)*	*End (side)*	*+z (xy plane)*
OE_2	$FA_{2\sim4,7,8}$	*End mill*	*Side*	*+z*
	$FA_{5\sim6}$		*Side*	
	FA_9		*End*	
OE_3	FA_{10}	*Drill*	----	*+z/-z*
OE_4	FA_{10}	*Boring tool*	----	*+z/-z*
OE_5	FA_{11}	*End mill (side mill)*	*End (side)*	*-x (yz plane)*
OE_6	FA_{12}	*Drill*	----	*+x*
OE_7	FA_{13}	*Drill*	----	*+x*
OE_8	FA_{14}	*End mill (side mill)*	*End (side)*	*-y (xz plane)*
OE_9	FA_{15}	*End mill (side mill)*	*End (side)*	*-z (xy plane)*

Determining Workpiece Locating and Clamping Methods for Each Operation Element

According to the principle of coincidence of references [196], design references specified in the design dimension/tolerance scheme should be used in the process planning when possible. However, if the design references are not suitable, other faces need to be selected. In this case, alternative dimensions and tolerances have to be calculated to meet the design specifications ([156], [157], Thimm et al. [198]). The selected locating faces must have sufficient area to accommodate the locating elements and must be sufficiently accurate. This constrains the machining sequence between a face and its locating references.

For each selected locating method, suitable clamping faces CFA_{FA} on the workpiece must be determined as well. The selected clamping faces must have sufficient area and stability to keep the workpiece in its locating position by the clamp. For the example part, the locating and clamping faces for each operation element are listed in Table 5.4.

Table 5.4 Locating and clamping faces for each operation element

Operation elements	*Faces to be machined*	*Cutter orientations (part coordinate system)*	*Locating faces*	*Clamping faces*
OE_1	FA_1	+z (xy plane)	FA_{15}	FA_{11}-FA_{16} or FA_{14}-FA_{17}
OE_2	$FA_{2\sim4,7,8}$, $FA_{5\sim6}$	+z	FA_{11}, FA_{14}	FA_{14}-FA_{17}
	FA_9		FA_{15}	
OE_3	FA_{10}	+z/-z	FA_{11}, FA_{14}	FA_{11}-FA_{16} or FA_{14}-FA_{17}
OE_4	FA_{10}	+z/-z	FA_{11}, FA_{14}	FA_{11}-FA_{16} or FA_{14}-FA_{17}
OE_5	FA_{11}	-x (yz plane)	FA_{16}	FA_1-FA_{15} or FA_{14}-FA_{17}
OE_6	FA_{12}	+x	FA_{11}, FA_1, FA_{14}	FA_1-FA_{15} or FA_{14}-FA_{17}
OE_7	FA_{13}	+x	FA_{11}, FA_1, FA_{14}	FA_1-FA_{15} or FA_{14}-FA_{17}
OE_8	FA_{14}	-y (xz plane)	FA_{17}	FA_1-FA_{15} or FA_{11}-FA_{16}
OE_9	FA_{15}	-z (xy plane)	FA_1	FA_{11}-FA_{16} or FA_{14}-FA_{17}

Arranging the Machining Sequence of Operation Elements

Machining constraints influence the feasible (or optimal) sequence of operation elements [83]. These constraints are defined as the machining sequence rules. Defining and selecting different machining sequence rules reflect a process planner's strategies (pursuing high quality, low cost, or short machining time, etc.). Different machining sequences result in different process planning features. For the example part, the applied machining sequence constraints are listed in Table 5.5. No constraint to the sequence of OE_5 and OE_8 as well as OE_6 and OE_7 exists.

Table 5.5 Machining sequences and the corresponding machining constraints

Machining sequences	*The corresponding machining constraints*
OE_9 *is before* OE_1 *and* OE_2 OE_1 *is before* OE_6 *and* OE_7 OE_5 *is before* OE_2, OE_3 *and* OE_4 OE_8 *is before* OE_2, OE_3, OE_4, OE_6 *and* OE_7	*The reference face must be machined before the faces referring to it.*
OE_2 *is before* OE_3 *and* OE_4	*To avoid generating burrs on the inside of the hole*
OE_3 *is before* OE_4	*Rough operations prior to finishing operations*

Generating Setups by Grouping Operation Elements

Each time a workpiece is re-located and re-clamped (a new setup used), the total machining time is increased and often the accuracy of the final part is decreased. Hence, the number of setups should be minimal. For each machine tool, operation elements that have the same cutter orientation, locating and clamping methods are grouped into a single setup *S*. For the example part, four machining setups are generated. The respective operation elements, machined faces, cutter orientations, locating and clamping faces are listed in Table 5.6. In this step, the cutter orientations are given with respect to the selected machine tool's coordinate system.

Table 5.6 Determination of setups

Machining setups	*Operation elements*	*Faces to be machined*	*Cutter orientations (machine tool coordinate system)*	*Locating faces*	*Clamping faces*
S_1	OE_5	FA_{11}	+z	FA_{16}	FA_1-FA_{15}
	OE_8	FA_{14}		FA_{17}	
S_2	OE_9	FA_{15}	+z	FA_1	FA_{11}-FA_{16} or FA_{14}-FA_{17}
S_3	OE_1	FA_1	+z	FA_{15}	FA_{14}-FA_{17}
	OE_2	$FA_{2\sim4,7,8}$, $FA_{5\sim6}$		FA_{11}, FA_{14}	
		FA_9		FA_{15}	
	OE_3	FA_{10}		FA_{11}, FA_{14}	
	OE_4	FA_{10}		FA_{11}, FA_{14}	
S_4	OE_6	FA_{12}	+z	FA_1, FA_{14}	FA_{14}-FA_{17}
	OE_7	FA_{13}		FA_1, FA_{14}	

Arranging the Sequence of Setups

The sequential relations between operation elements of different setups determine the sequence of setups. If the sequence of two setups contradicts the sequence constraint of any operation elements, the offending operation elements are re-assigned to other setups or new setups need to be generated. Due to the machining sequence constraints listed in Table 5.5, the setup sequence for the example part is (S_1, S_3, S_4) and (S_2, S_3). There is no constraint on the sequence of S_1 and S_2.

Determining Machining Parameters for Each Machining Cut

The machining parameters for each machining cut include the cutting speed, depth of cut, feed rate as well as the intermediate dimensions and tolerances. The determination of these parameters is involved (Halevi & Weill [199]). Major influence factors include the selected machining routes, the required tolerance, surface roughness, allowances for the finish operations, the shape and size of the blank, machining cost constraints. These parameters are usually selected from the machining handbooks.

Process Planning Feature Generation

The final step of process planning is the generation of the feature description. That is a set of interrelated process planning features. The data recorded in the operation elements (machining method, cutter type, cutter section and orientation) determine the basic shape and orientation of the corresponding process planning features. The machining sequence of operation elements and the machining cut division in a single operation element determine the dimensions of the process planning features. Figure

5.11 illustrates the process planning feature description of the example part. Clearly, process planning features are not simply geometric volumes. They are associated to other process plan specific entities, such as machine tools, cutters, and setups. The process planning features are determined by machining constraints (represented as process planning rules). The machining constraints may be hard (e.g. the reference constraint) or soft (generated from the consideration of machining cost/time optimization, such as minimizing the numbers of setups or cutter changes).

Firing different rules could result in different selections of the machining methods, cutters, machining sequences, and machining parameters, which further determine the type, number as well as the shapes or dimensions of the generated process planning features in the process planning feature model. The geometry (shape, dimension, position, and orientation) of a process planning feature is directly determined by the generated setups, operation elements, their sequences, and the intermediate dimensions for each machining cut.

5.2.3 Rule-Based Constraints in Process Planning Stage

The input to this stage is the detailed design. The output is a workable process plan. Based on the above analysis (about the inherent relations in this stage), rule-based constraints are used in machining method selection, machining operation feasibility evaluation, machining sequence determination, and machining cost restriction. The format of rule definition is similar to those in Section 5.1.4.

5.3 Summary

The importance of integrating the conceptual design stage into the whole product modeling system is analyzed. Two sub-types of the unified feature class, conceptual design feature and process planning feature, are defined in this chapter. The generation processes of these two types of features are described in detail. It can be seen that application features are tightly linked to the engineering intent of using them. Pure geometric definitions are insufficient. Engineering knowledge as well as other related entities must be associated with the application features and should be consulted in the feature validation processes.

In addition, embedding engineering intent into product models using KBE, using engineering knowledge to drive product modeling or process planning, associating

engineering knowledge with product designs or process plans, using engineering intent to justify modifications, and searching alternative solutions can be achieved using the proposed approach.

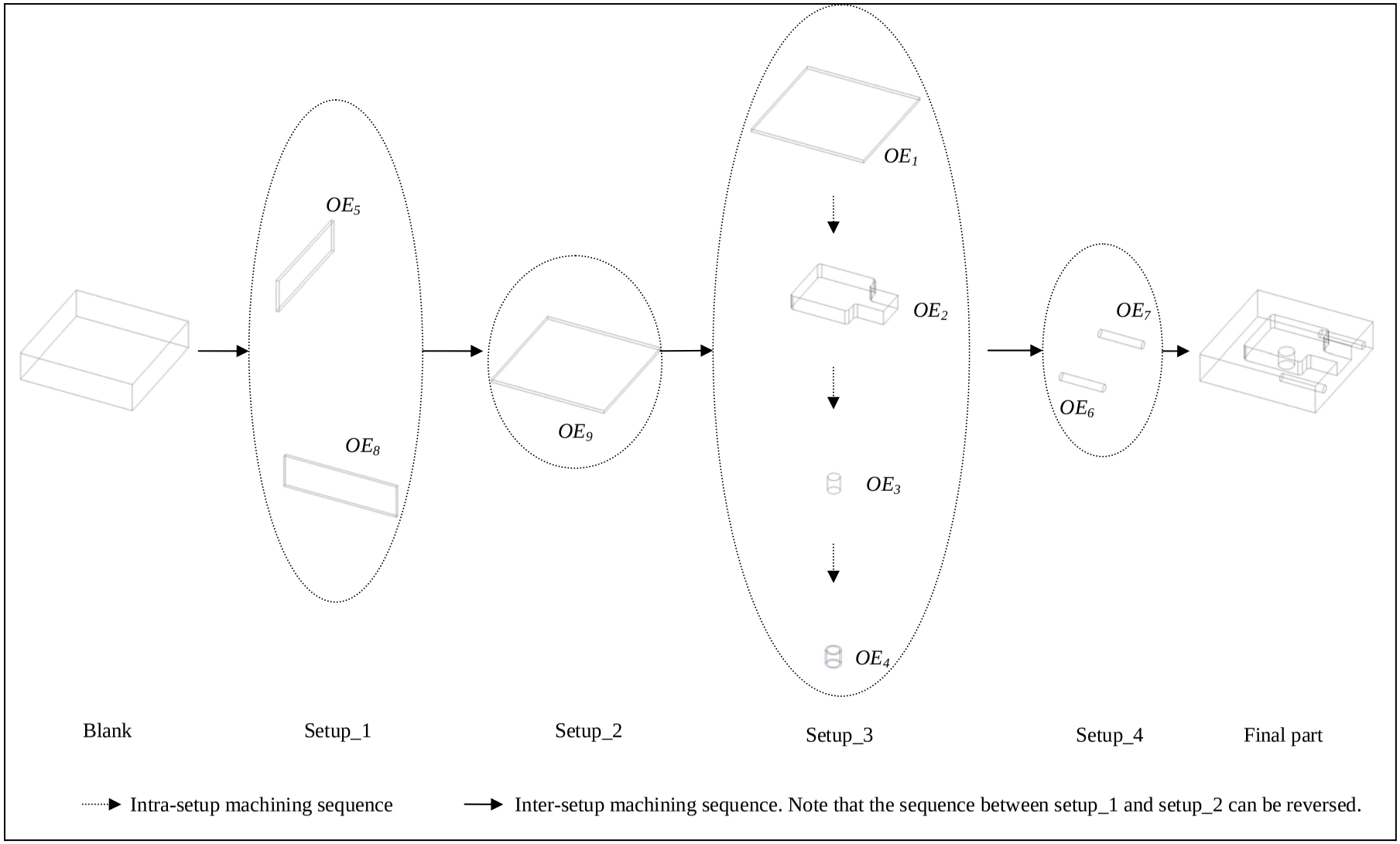


Figure 5.11 Process planning feature description of the example part

Chapter 6
Implementation of Associations

6.1 Introduction

The utmost purpose of the unified feature-based product modeling scheme is to maintain the validity, consistency, and integrity of product models. As discussed in previous chapters, traditional CAx systems have limitations to serve this purpose. Two major problems are:

- Engineering intent is not well represented and managed.
- Inter-stage, non-geometric relations are not well-maintained.

The unified feature-based product modeling scheme tackles these two problems via establishing and maintaining geometric and non-geometric data associations, within a single or across different stages. These associations are generalized as constraint-based associations and sharing associations (Figure 6.1).

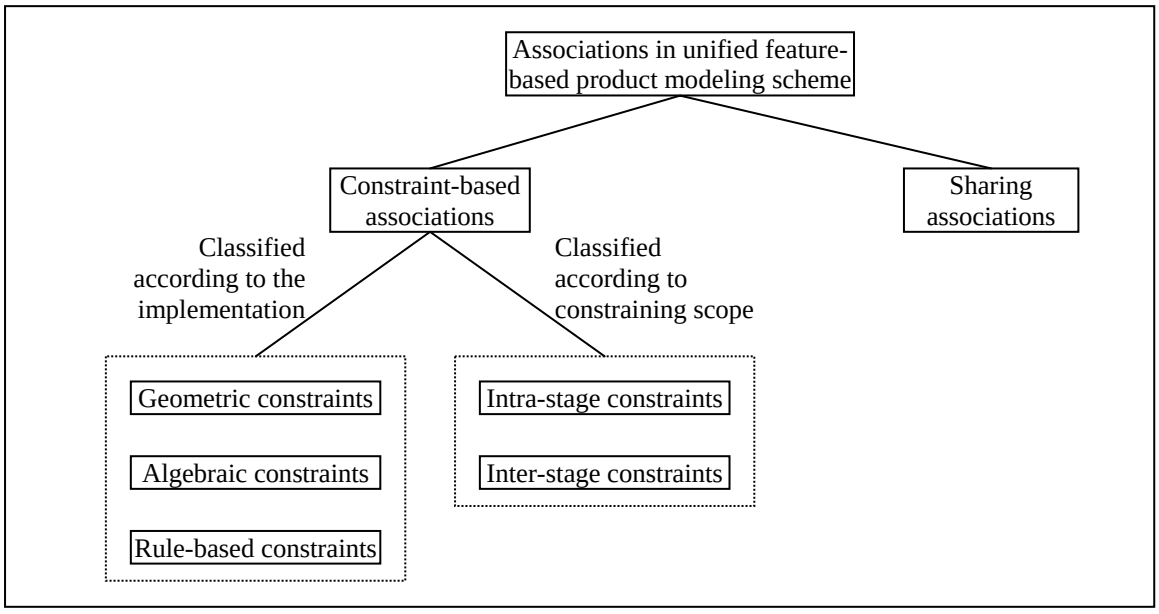


Figure 6.1 Associations in the unified feature-based product modeling scheme

Constraint-based associations are established on the basis of intra- or inter-stage, numerical or rule-based constraints. Sharing associations are established based on the unified cellular model. In this chapter, the implementation of these two types of associations is described first (Section 6.2 and 6.3). Next, a set of criteria, which is used

to evaluate the validity and integrity of a unified feature-based product model, is given in section 6.4. A change propagation algorithm is described in Section 6.5.

6.2 Implementing the Constraint-Based Associations

Together with a rule-based expert system and a numerical constraint solver, a Justification-based Truth Maintenance System (JTMS) is used to implement constraint-based associations. A JTMS dependency network consists of a series of related nodes which represent the belief status of entities. *Assumption* nodes are believed without any supporting justifications. *Simple nodes* are only believed if they have valid justifications. An assumption node can be converted into a simple node, which then needs to be supported by justifications. A justification consists of antecedent nodes and consequent nodes. A node is said to be justified by a supporting justification if all antecedents of the justification are justified.

Whenever a constraint-based association is generated, the corresponding JTMS nodes and justifications are inserted into a JTMS dependency network. After the insertion process, each node records three items:

- A reference to its direct supporting justification;
- References to the justifications that use this node as antecedent (for later change propagation);
- Its current belief status.

In Figure 6.2, the black triangles indicate *premise nodes* that do not require a supporting justification and will never be revoked. The figure illustrates that the justification of node N_3 consists of nodes N_1, N_2, and R_3/C_3 (R for rule and C for constraint) while the justification of node N_1 is AS_1, AS_2 (AS for assumption), and R_1/C_1. Node N_3 is ultimately justified by AS_1, AS_2, AS_3, R_1/C_1, R_2/C_2, and R_3/C_3. Retraction of any one of these nodes invalidates node N_3.

Whenever a modification to the JTMS dependency network occurs, such as adding or retracting assumptions, modifying nodes or adding justifications, the JTMS dependency network is searched for the affected nodes as well as the related justifications. If it is a rule-based constraint to provide the justification, the system refers to the knowledge base to validate the modification. If it is a numerical constraint to provide the justification, the system refers to the numerical constraint solver to validate the modification. These checking and change propagating processes are automated. The result is a new status of

each affected JTMS node or a rejection of the modification on the basis of contradicting beliefs. The data structures and algorithms of JTMS are generic. Therefore, it handles geometric and non-geometric constraints uniformly.

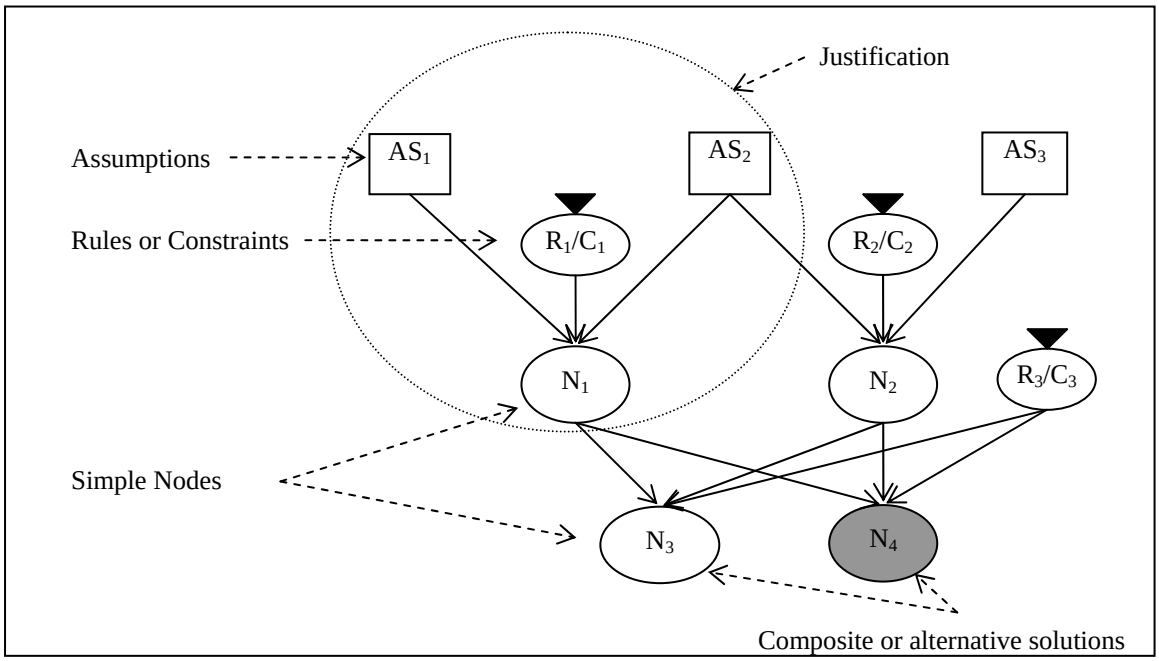


Figure 6.2 JTMS-based dependency network

A relational database is used for all applications to store and publish their data. An application can access and enquire the database for data published by other applications. When an inter-stage constraint-based association is established, this association and the involved data are stored in the database. When an application modifies its model, it must check the database for relevant inter-stage associations. If such associations exist, a validity checking process is triggered. The involved applications are responsible for maintaining the consistency (between associated stages) while the database is a medium for storing the repository data, inter-stage associations, and propagating changes. Figure 6.3 illustrates the constraint-based associations between the conceptual and the detailed design feature models. The constraint-based associations between the detailed design and the process planning feature models are established in a similar way.

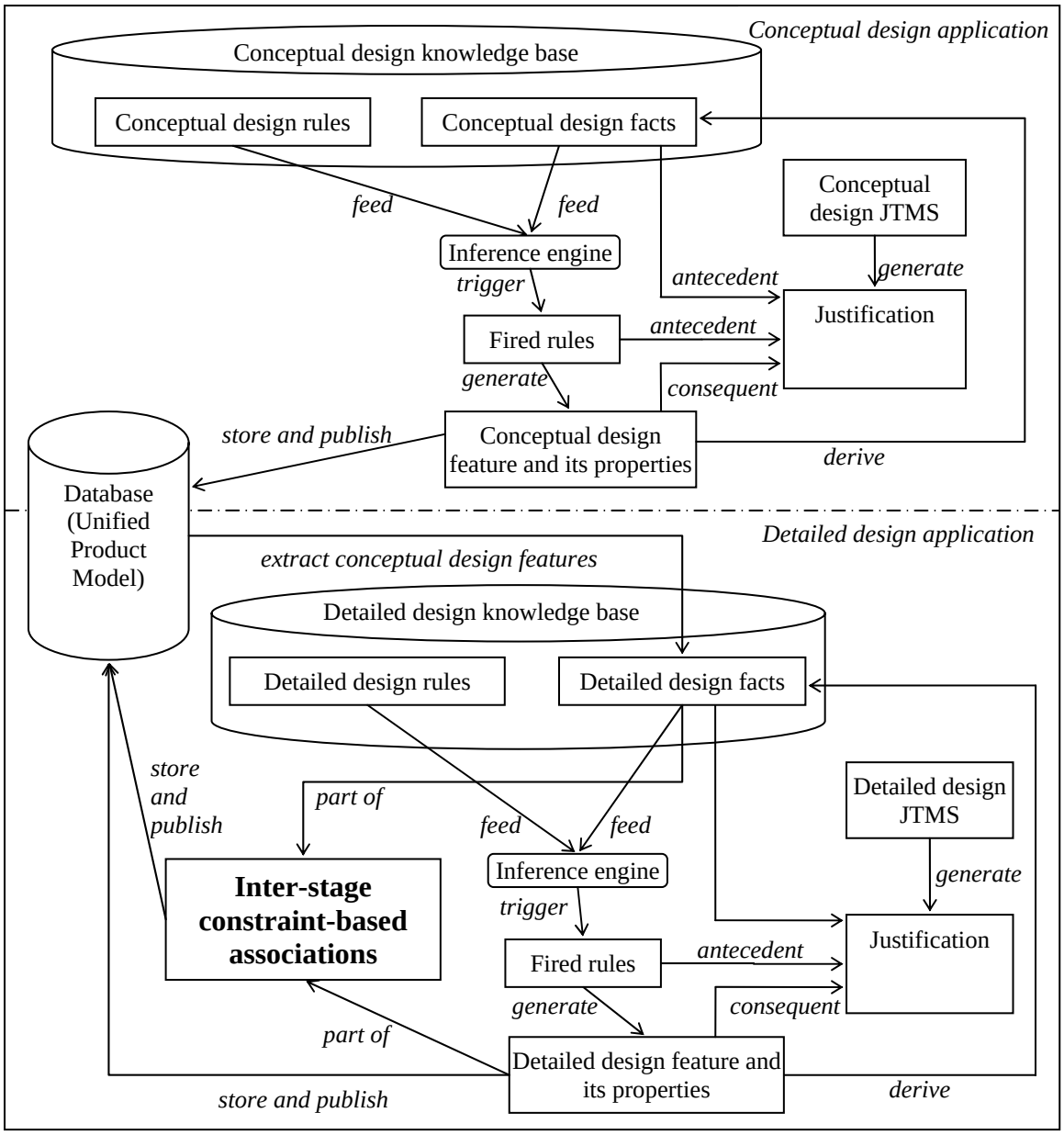


Figure 6.3 Constraint-based associations between conceptual and detailed design stages

6.3 Implementing the Sharing Associations

A unified cellular model (Chapter 4), which is stored in a database, is used to implement the sharing associations.

6.3.1 Generating a New Application Feature

Each application feature class has its geometry creation and manipulation functions. When a creation function is invoked, the feature geometry is created and inserted into the application's runtime cellular model. The created topological entities are associated with

the feature through the owning feature attributes and the feature's runtime geometric references. The feature geometry is also inserted into the unified cellular model. If any cell in the unified cellular model is affected by this new feature, e.g. overlapping, the owning features of the affected cells are marked for validity checking.

6.3.2 Modifying an Application Feature

When an application feature is modified, in addition to updating the application's runtime cellular model, the application also notifies the unified cellular model about the modifications. The unified cellular model is updated and the affected cells are marked as been modified. The owning features of the affected cells are then validated by the corresponding applications.

The sharing association mechanism enables different application features to be associated with the same geometric or topological entities and hence supports achieving inter-stage geometric consistency.

6.4 Evaluation of the Validity and Integrity of the Unified Feature Model

The general requirement for a valid product information model is that each application model (corresponding to a particular stage) must be valid and also consistent with other associated application models. The detailed evaluation criteria are classified into feature-, intra-stage-, and inter-stage-level.

- A feature is valid if:
  - (i) The feature geometry refers to valid topological entities.
  - (ii) The values of feature parameters are consistent with the product's geometric model.
  - (iii) All constraints on the feature are satisfied.
  - (iv) Any feature property, if included in the JTMS dependency network, has a "believed" status, i.e. its supporting justifications are valid.
- A product model is valid if:
  - (i) All features in the model are valid.
  - (ii) In its knowledge base, the antecedent conditions of all fired rules, which are the justifications for the generated features (or feature properties), are satisfied.
  - (iii) All constraint-based associations between consequent facts and respective features (or feature properties) hold.

(iv) Cellular entities, which are referred by the geometric references of all the existing features, exist and have the correct status (material or void, on the boundary or not on the boundary) according to the feature sequences in their owning feature lists.

- Two product models (corresponding to different life cycle stages) are consistent if:
  (i) Sharing associations between their corresponding application features hold.
  (ii) Constraint-based associations between their corresponding application features or feature properties hold. In particular:
    a. Each critical feature in the conceptual design is linked to features in the detailed design via valid constraint-based associations.
    b. Each feature property or inter-feature constraint in the conceptual design has its valid counterparts (may not be one to one relations) in the detailed design.
    c. Each detailed design feature to be machined is linked to process planning features via valid constraint-based associations.
    d. All the design specifications (such as tolerances and surface finishes) are satisfied by the finish process planning features.

6.5 Algorithms for Change Propagation

If users (designers or process planners) modify the product model, the modifications must be checked to make sure that the consistency of the whole product information model is maintained. As indicated in previous sections, a dependency network is established using constraint-based associations and sharing associations. It is implemented through a JTMS and a common database. The purpose of the dependency network is for the propagation of modifications and determining the influence scope of a modification.

The propagation and checking process is divided into two major steps: local checking within a specific model and global checking across different models (Figure 6.4). Assume variable 1 in application 2 is changed (as the initial modification). In the figure, arrows are directed from driving to driven variables while “var” represents variables.

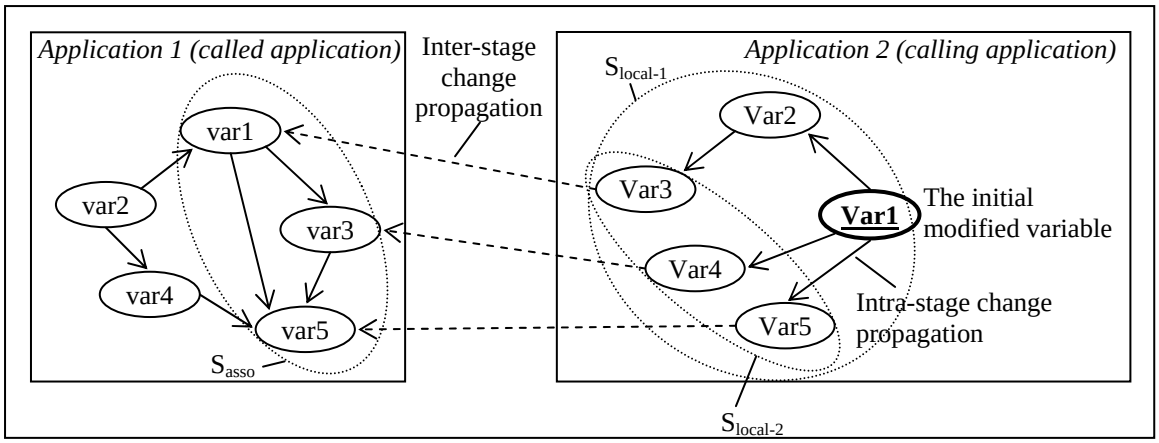


Figure 6.4 A chain of intra- and inter-stage change propagation

The change propagation algorithms are given as follows (for detailed pseudo code, please refer to Appendix A).

PROCEDURE **Check_Local(x)** */* checking the intra-stage associations */*

- Backup the value of the initial modified variable x. Put x into a local set (set_1, which records modified variables v_i that need to be checked for intra-stage associations). For each v_i in set_1, search the JTMS dependency network for variables that associate to v_i using JTMS attributes (antecedent or consequent). The variables, which are antecedents of v_i, are driving variables. The variables, which are consequents of v_i, are driven variables.
- Check the constraints between each v_i and its driving or driven variables one by one:
  (i) If the new value of v_i violates the constraints between v_i and any of its related variables:
    - If the related variable is a driven variable
      a. If the value of the driven variable is fixed by the constraint, i.e. without alternative values, then the modification is rejected. Abandon().
      b. If the driven variable has alternative values, search one for which the constraint is satisfied:
        - If the constraint can be satisfied (and the value of the driven variable is changed), make a backup of the old value and put the driven variable into set_1.
        - If no alternative value satisfies the constraint, then Abandon().
    - If the directly related variable is a driving variable, then Abandon().

(ii) If no constraint has been violated or if some constraints has been violated but can be re-satisfied, the modification is locally accepted. Then, further checking is carried out for v_i in the database. If v_i appears in any inter-stage associations in the database, move v_i from set_1 to set_2 (which records variables that need to be checked for inter-stage associations). Check_Global().

PROCEDURE **Check_Global()** */* checking the inter-stage associations */*

- For each member of set_2, add all associated features or feature properties in the database to set_3 (which records associated variables in other applications). An initial modification in an application may invoke many modifications in other applications (see Figure 6.4). The members of set_3 are checked (in the next two steps) one by one until set_3 becomes empty.
- The values of members of set_3 are temporarily changed in the database using the constraints recorded in the calling application.
- For each member of set_3, execute Check_Local() in the called application until the modification is found to be locally accepted or rejected. The corresponding message (about whether the modification is locally accepted or rejected in the called application) is sent back to the calling application that initiates the initial modification.
  (i) If all modifications are accepted, the initial modification in the calling application is globally accepted and committed.
  (ii) If any of the modifications are rejected, the initial modification in the calling application is rejected, then Abandon().

PROCEDURE **Abandon()** */* retracting all changes temporarily made */*

- All modifications made in the calling and called applications are revoked using backup values.
- In the database, the data of the called application, whose values are temporarily changed, are set back to their previous valid values.

6.6 Summary

In this chapter, constraint-based and sharing associations as well as their corresponding implementations in the unified feature-based product modeling scheme

are first discussed. Then, using a JTMS and a common database to establish inter-stage non-geometric associations is illustrated. Finally, the criteria of evaluating the validity of product models and an algorithm for change propagation are proposed.

With the unified feature definition, application feature definitions, rule-based expert system, the unified cellular model, dependency network, and the change propagation algorithm, the proposed unified feature-based product modeling scheme is able to integrate the conceptual design, detailed design, and process planning applications. Three case studies are used to illustrate the feasibility of the proposed scheme in the next chapter.

Chapter 7

Prototype System and Case Study

7.1 Prototype System

A prototype system was developed to demonstrate the capability and feasibility of the proposed unified feature-based product modeling scheme. Three applications, conceptual design, detailed design, and process planning, are included in the prototype system.

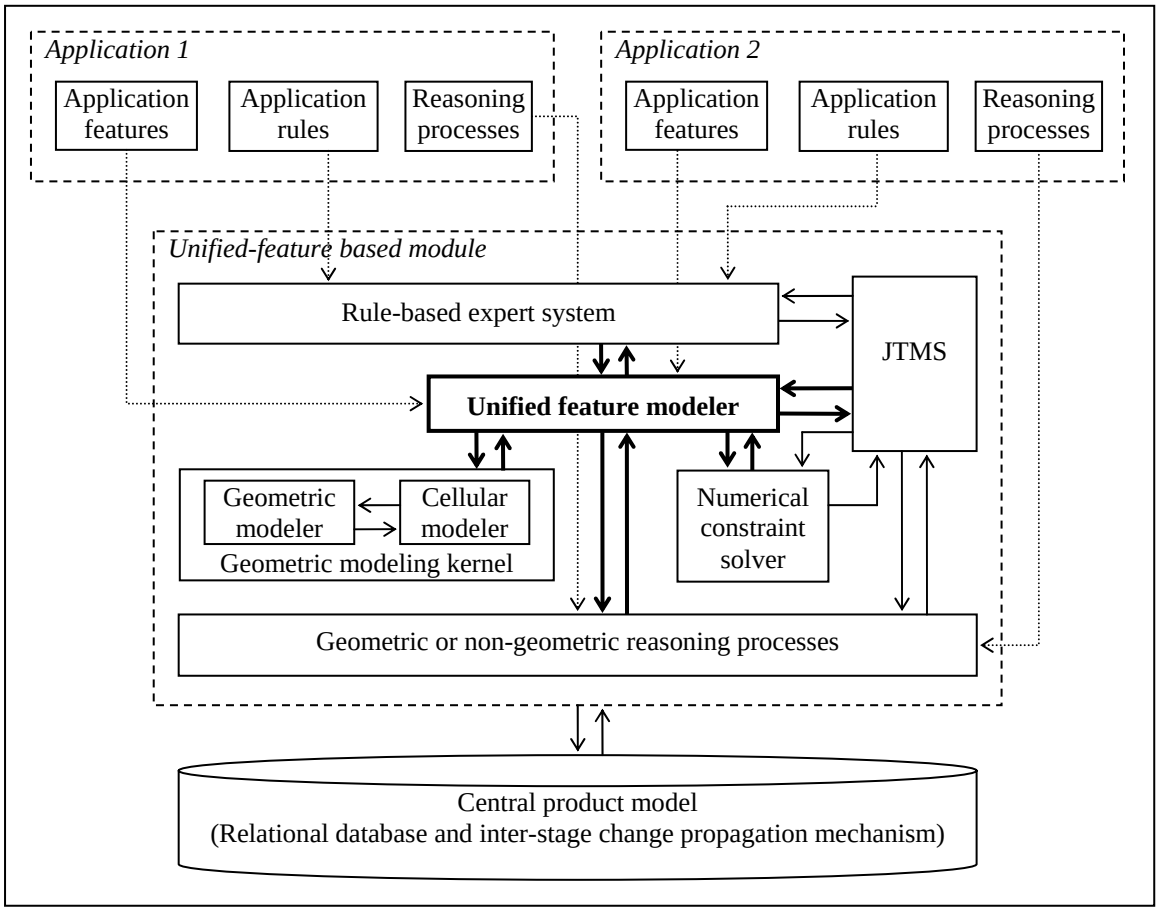


Figure 7.1 Structure of the prototype system

As shown in Figure 7.1, in the implementation, these three applications are supported by a common unified feature based module, which includes a rule-based expert system, a JTMS, a unified feature modeler, a geometric modeling kernel, and a numerical constraint solver. Each application also has its application specific features, rules, and other entities. In addition, applications store and publish data in a database accessible by

other applications. This prototype system is implemented with Visual C++ 6.0 and ACIS 7.0 (ACIS [200]) as the foundational building blocks. In the following sub-sections, the main components of the unified feature-based product modeling scheme, which are feature modeler, rule-based expert system, JTMS, and the database, are described.

7.1.1 Feature Modeler

The geometric modeling functions are realized by an ACIS geometric kernel and a numerical constraint solver. Common form features (such as plane, block, cylinder, hole, pocket, and sweeping), feature attributes, and constraints are all defined as ACIS entities using the API methods provided by the ACIS kernel. Although each application has its specific features, the common fields and methods of these application specific features are defined in the unified feature class directly using the ACIS methods.

The cellular topology component provided by the ACIS kernel is used in the cellular model. Figure 7.2 shows that the feature class is supported by many classes defined in the ACIS kernel. Two- and three-dimensional cells are supported in the current prototype system. Feature-level operations, such as editing or displaying, are based on the cellular model. The boundary representation is derived from the cellular model.

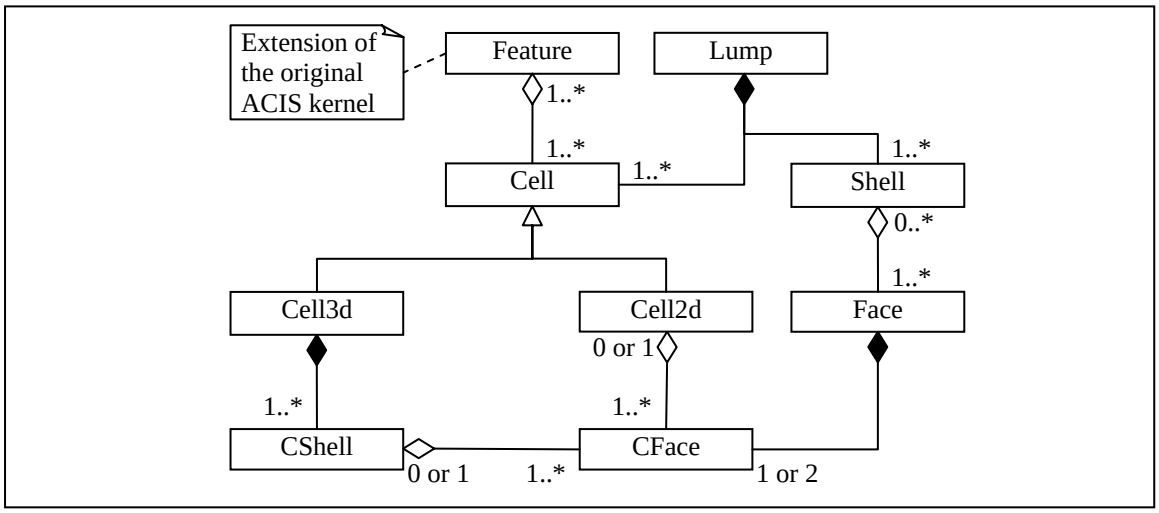


Figure 7.2 ACIS-based cellular topology for feature modeling

The cellular model is linked to the feature model through cells' owning feature attributes (refer to Section 4.3.1). The cell data is attached to LUMPs. A cell links to faces (in the boundary representation) through cell faces (CFACE). Since ACIS only

attaches cell data as attributes to normal ACIS topology (refer to ACIS documentation [200]), additional functions developed in the prototype system include:

- Managing a cell's attributes on nature and owners during the manipulation of features.
- Maintaining the consistency of the runtime application cellular models and the unified cellular model (stored in a database).

The consistency relations between the features, cellular models, and the boundary representation are maintained. These relations are the basis of the sharing associations.

A simple numerical constraint solver was developed to solve algebraic and geometric constraints. The numerical constraint solver communicates with the ACIS kernel to get the necessary information and feeds back the caller about the validity of the constraints, i.e. whether a checked constraint is satisfied or not. The numerical constraint solver can also solve the specified constraints, with the new values of the constrained feature properties, in order to re-satisfy the violated constraints. The relations between the numerical constraints and the related feature properties are recorded as constraint-based associations for supporting enquiries and modifications via appropriate methods when necessary.

7.1.2 Rule-Based Expert System

The module implementing the forward-chaining rule-based expert system is common to all applications, although each application has its specific rules and facts. Major classes have been shown in Figure 4.4. The algorithm of pattern matching is shown in Figure 7.3. This algorithm finds out all rules that are ready to fire. These rules are then fired one by one and new facts are generated respectively as consequences. Rules can be predefined, stored in a database, and loaded into a session at runtime. Through the prototype system, rules can also be generated interactively as given by a user. New facts are generated during the modeling process. For example, newly created features or constraints derive new facts, which are inserted into the fact base. Each fact is linked to its owning entity through recording the entity's unique identifier. The relation between a fact and its corresponding owner as well as the relation between the antecedent pattern and the consequent pattern of a rule, are recorded using JTMS for enquiries and modifications.

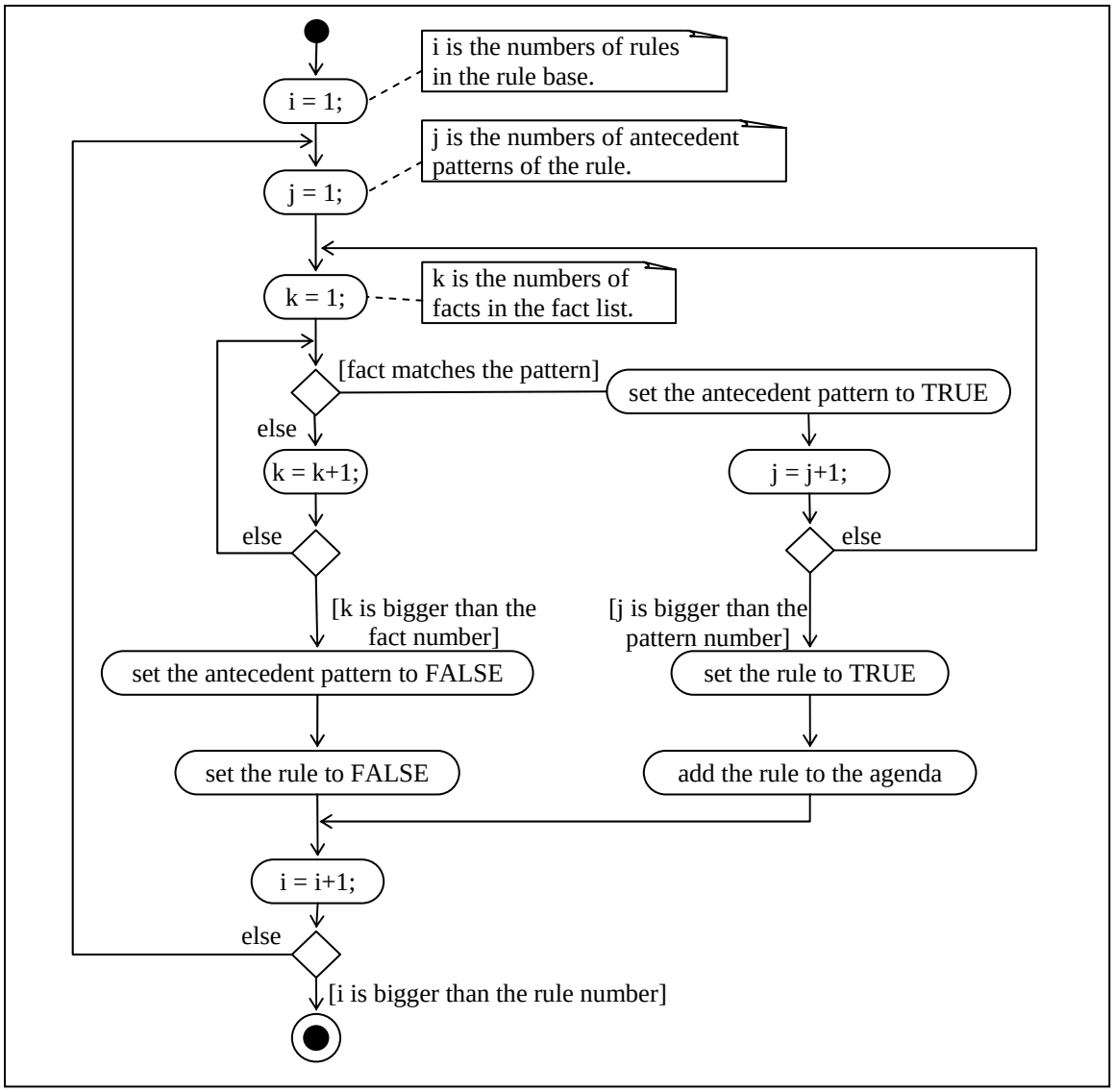


Figure 7.3 Pattern matching algorithm of the rule-based system

7.1.3 Justification-Based Truth Maintenance System (JTMS)

A JTMS is developed based on [178] to record the constraint-based associations generated along with the product development process (Figure 7.4). The entities (features, feature properties, constraints, rules, facts, and so on) involved into a constraint-based association are represented as nodes individually in the JTMS dependency network. Each node records the identifier of its owning entity. The JTMS is mainly responsible for recording and updating the truth status of each node in the dependency network, i.e. it provides the belief status of each entity to the inference engine. The inference engine drives the modifications of the truth status of nodes in the JTMS through adding justifications, enabling or retracting assumptions. In general, the

JTMS coordinates with the inference engine (a numerical constraint solver and a rule-based expert system) to provide the problem solving capabilities of the prototype system.

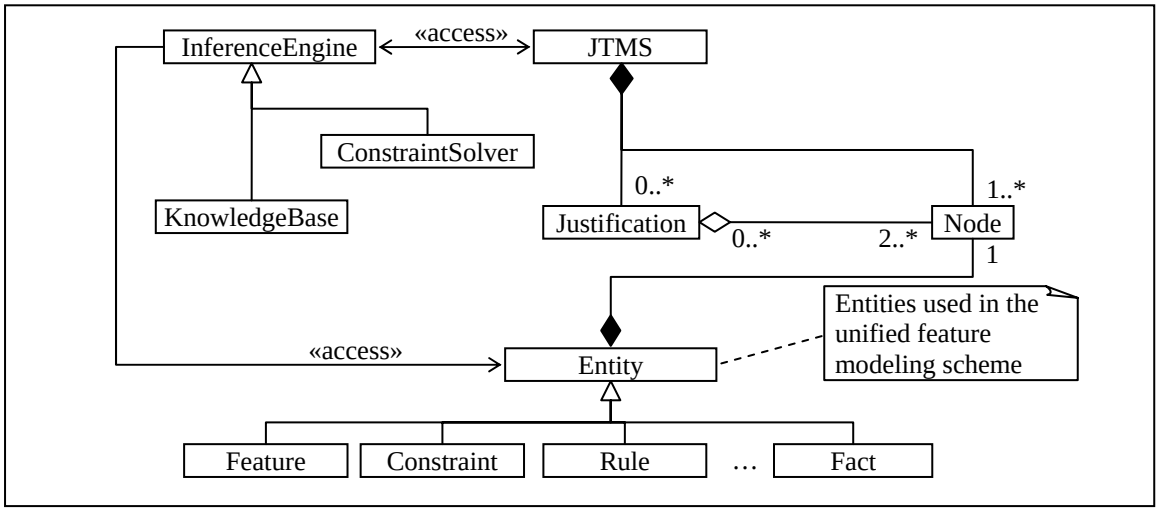


Figure 7.4 The inference engine and the JTMS in a problem solver

7.1.4 The Relational Database

Each application has its specific feature classes that other applications cannot access directly. Different applications use a relational database as the medium to share data with each other. In the prototype system, this database is developed using MySQL 5.0.1, (MySQL [201]) and is used to store three major kinds of data:

- Unified cellular model. An independent application is developed to update this unified cellular model according to the newly generated features or feature modifications.
- Data published by each application.
- Inter-stage associations.

Each related application is responsible for maintaining the vadidity of its own entities and associations. The database is just a medium for storing common information, inter-stage associations, and for propagating changes. The schema diagram of the relational database is partially shown in Figure 7.5. Schemas for detailed features are left out for clarity.

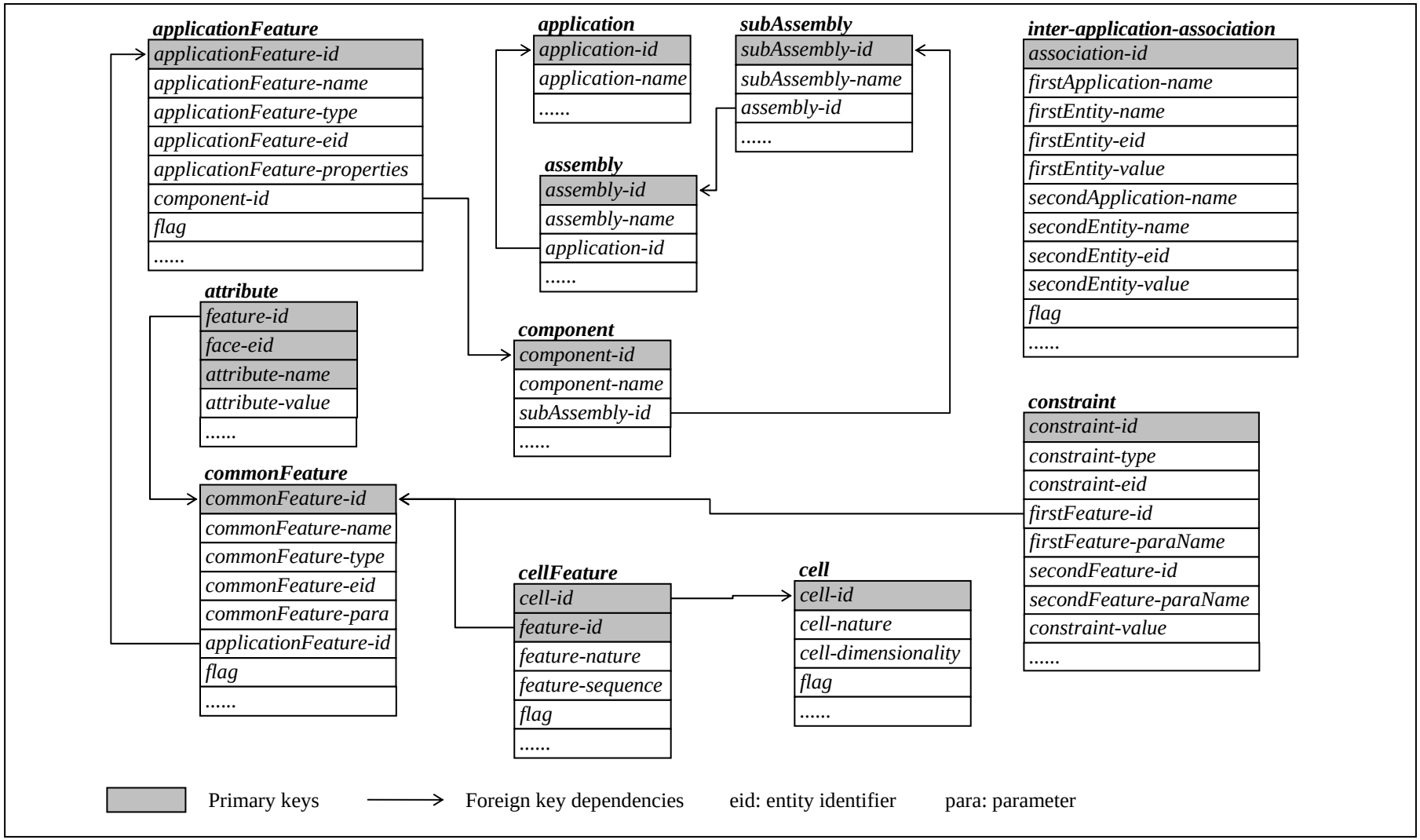


Figure 7.5 Partial schema diagram of the relational database

7.2 Case Study

Three cases are used to illustrate the proposed unified feature-based product modeling scheme and to demonstrate the prototype.

7.2.1 Association between Conceptual and Detailed Design Stages – Case 1

The first case study discusses the development process of an ejection system in a plastic injection mould. It is intended to illustrate the association and change propagation within and between the conceptual and detailed design stages.

Function Decomposition and Feasibility Evaluation

The conceptual design process starts from the overall function “create the mould”. This overall function needs to be decomposed manually by the designer (Figure 7.6).

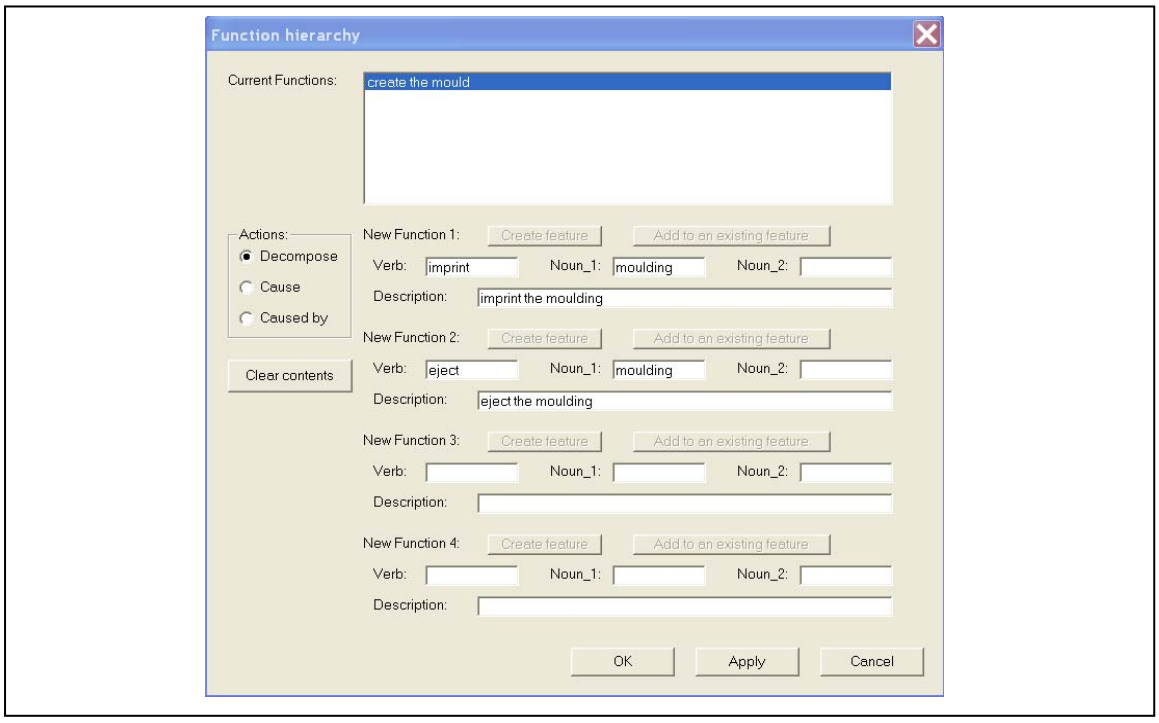


Figure 7.6 Function hierarchy dialog

Here, the designer decomposes it into two sub-functions “imprint the moulding” and “eject the moulding”. Since “imprint the moulding” is a primitive function, it is not decomposed further. Eventually, this function will lead to core and cavity design features including different sub-inserts. However, “eject the moulding” is not a primitive function, and is further decomposed into three primitive sub-functions: “imprint the

moulding in the contact areas of the ejector", "propel the moulding", and "guide the ejector". In addition, an auxiliary function, "prevent moulding damage", is created along with the "propel the moulding" function. This auxiliary function imposes an algebraic constraint between the wall thickness of the moulding and the number of the ejectors. This is necessary to limit the pressure that each ejector exerts on the moulding. The system automatically transforms the designer's specifications as new functional design rules.

Function-to-Feature Mapping and Critical Property Specification

The system then generates a corresponding conceptual design feature for each primitive function. The designer may either specify the feature properties or use the default values (Figure 7.7).

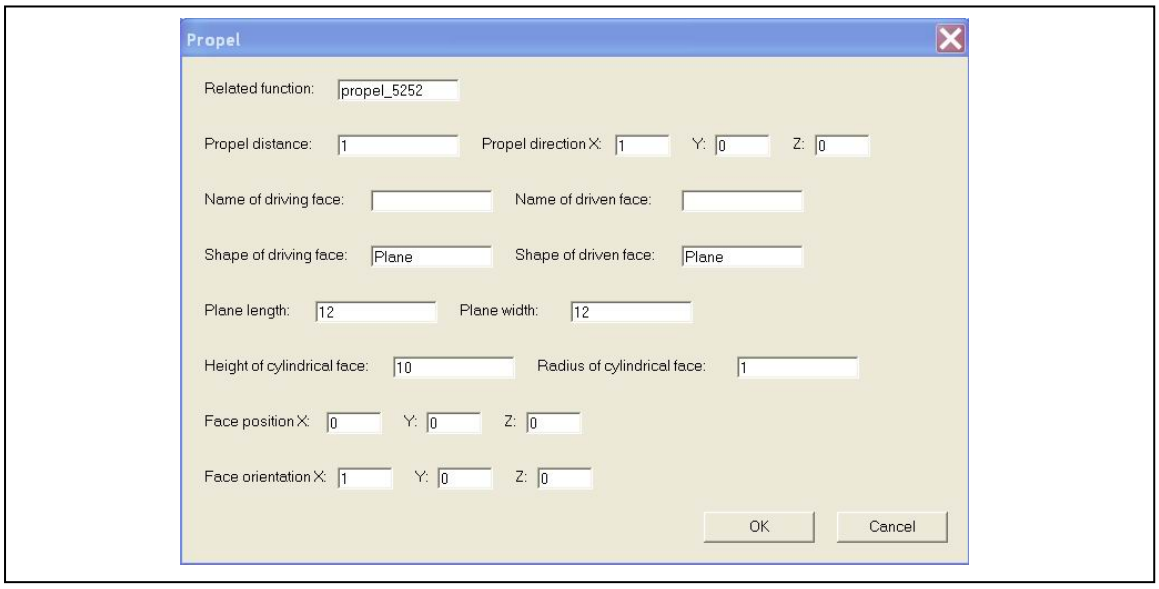


Figure 7.7 A propel feature

Figure 7.8 shows that three conceptual design features ("imprint", "propel", and "guide") are generated. Their corresponding properties and constraints are generated respectively. Since their geometries are unspecified, the symbolic forms of features are used.

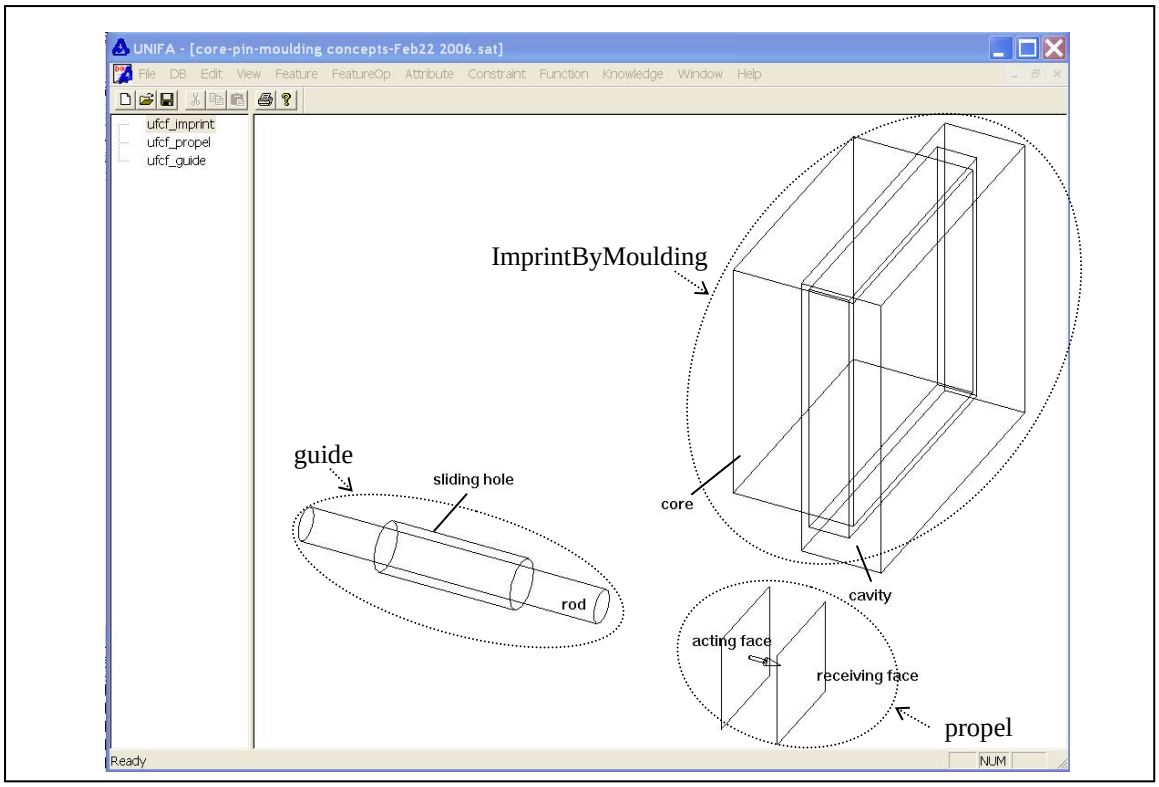


Figure 7.8 Conceptual design feature model

After the newly generated “propel” feature is transformed into a new fact and inserted into the knowledge base, a critical feature property rule (specifying the type of the ejector) fires. The rule determining the ejector type is predefined as follows:

IF (*FeatureName* is “propel”) AND (*DrivingEntity* is “ejector”)

THEN (*EjectorType* is (normal ejector pin) OR (D-shape ejector pin) OR (stripper plate))

Since the consequent part of this rule has alternative options, the designer needs to make a selection. Other options are kept with the fired rule for future enquiries. The conceptual design application publishes its features into the database.

All decomposition and sequential relations between functions, mapping relations between functions and features, rules and constraints, are recorded as associations using the JTMS (Figure 7.9).

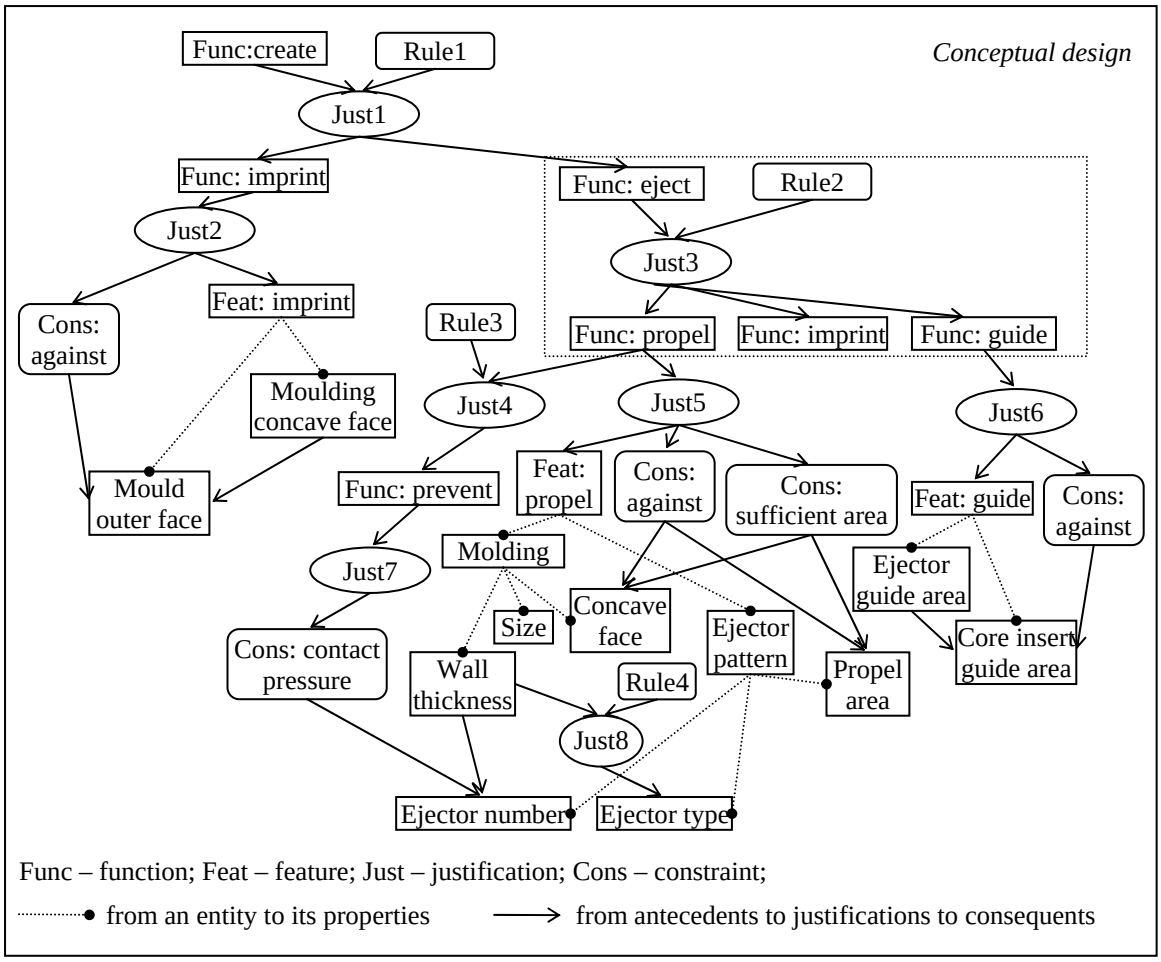


Figure 7.9 The JTMS dependency network for the conceptual design of an ejection system

Figure 7.9 shows the dependency network established during the conceptual design of the ejection system. Plain squares represent entities, such as functions, features, or feature properties. Squares with round corners represent constraints or rules. Circles represent justifications. Arrows are directed from antecedents to justifications, and further to consequents. In the figure, the portion surrounded by the dashed square represents a primitive reasoning process: based on the input function "eject the moulding" and "Rule2" as antecedents, justification "Just3" generates three sub-functions "imprint the moulding in the contact areas of the ejector", "propel the moulding" and "guide the ejector" as the consequents. All dependency networks in this chapter are created in the similar manner. Please also see Figure 6.2 for explanation.

Detailed Design of the Ejection System

After entering the detailed design stage, the designer gets the conceptual design via the user interface (Figure 7.10), which makes enquiries to the database, and gets the necessary information, such as the conceptual design features.

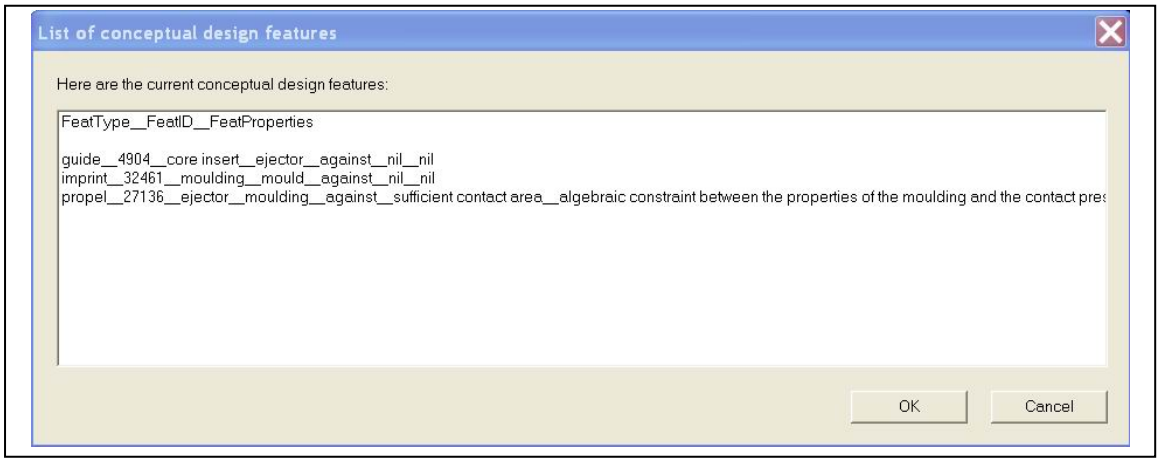


Figure 7.10 Reading the conceptual design in a detailed design application

When the designer starts the detailed design of the ejection system according to the characteristics of the moulding, detailed design features are created based on the conceptual ones and other design considerations. In the case shown in Figure 7.11, the moulding is a simple box-type moulding with a spherical top shape.

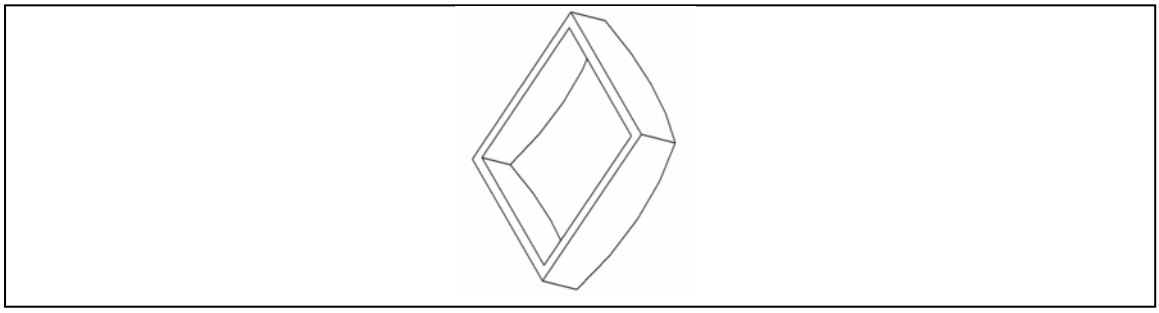


Figure 7.11 The original moulding

The detailed design for the ejection assembly is shown in Figure 7.12. The assembly includes several components, such as core insert, core plate, central ejector pin pattern, boarder ejector pin pattern, retaining plate, and ejector plate.

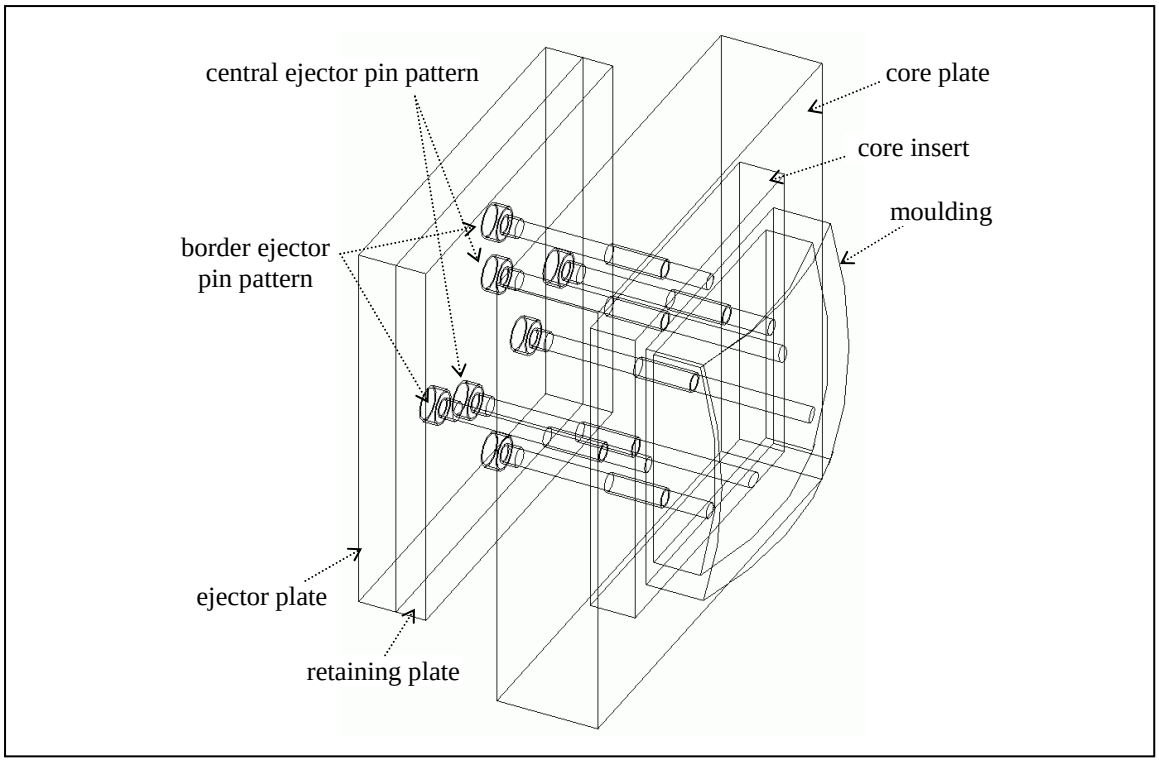


Figure 7.12 The detailed design assembly of the ejection system

The central ejector pins are responsible for ejecting the central portion of the moulding while the border ejector pins are responsible for ejecting the wall sections of the moulding (Figure 7.13). Some numerical constraints are generated during the detailed design process:

- The shrinkage relations between surfaces of the moulding and those of core or cavity inserts;
- "Completely adjacent" constraints between the faces of the core insert and the head faces of ejector pins;
- Co-axis constraints between the ejector pins and the corresponding holes in the core insert;
- An algebraic constraint on the contact area between moulding walls and the head faces of the ejector pins;
- Another algebraic constraint on the size of the moulding and the number of ejector pins.

These constraints are all recorded using the JTMS.

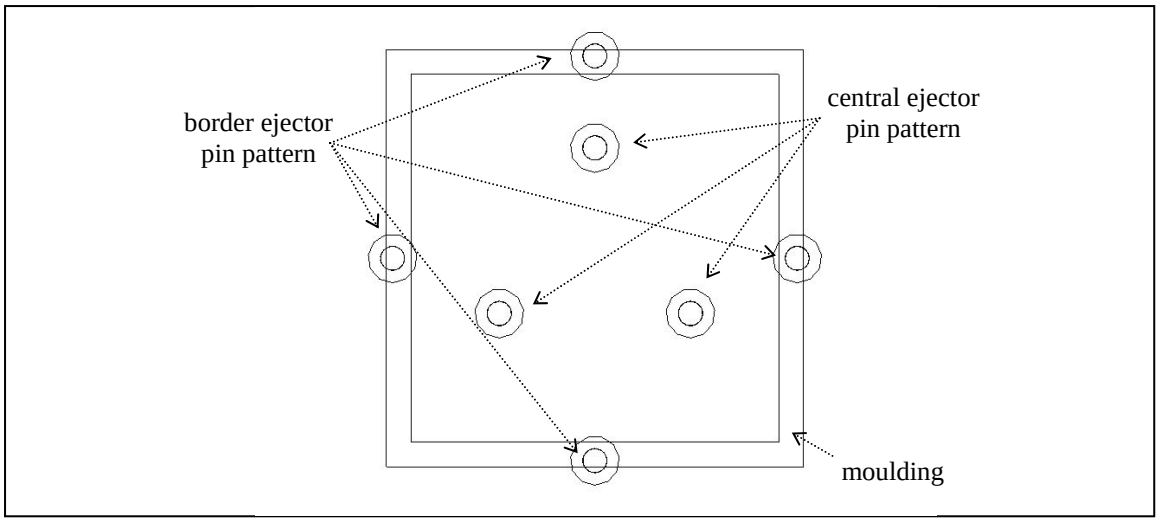


Figure 7.13 Border and central ejector pin patterns

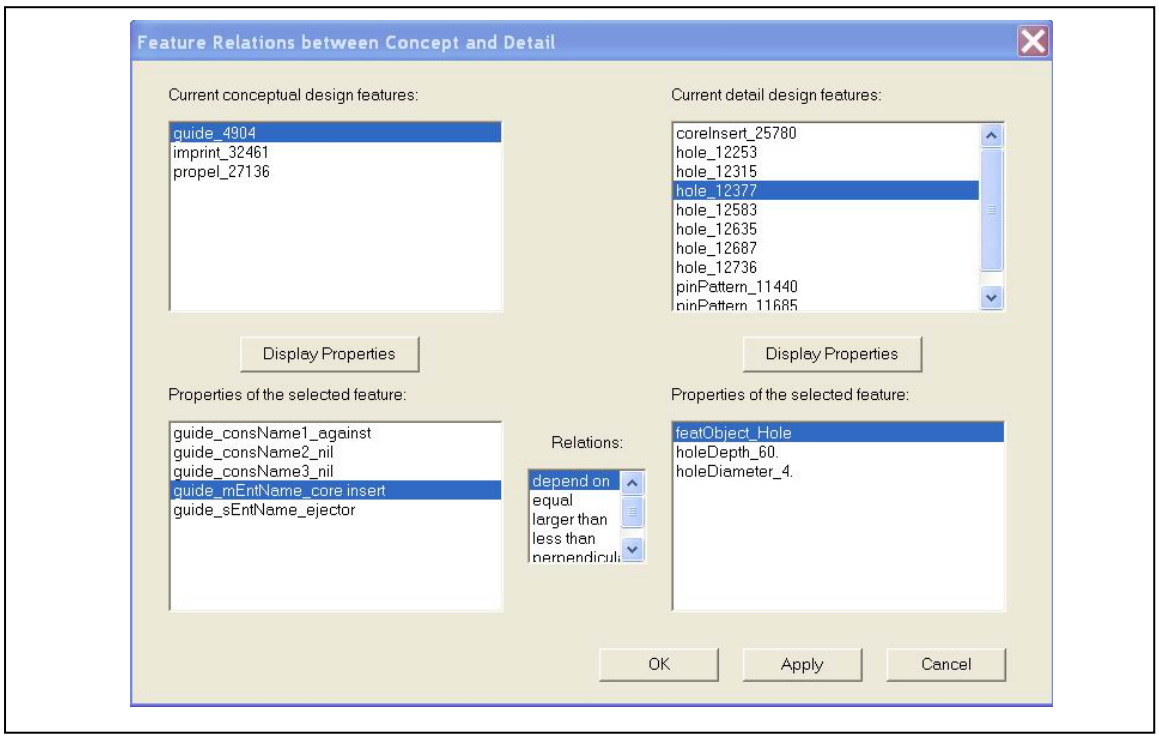


Figure 7.14 Associating features in conceptual and detailed designs

Some detailed components, features, constraints correspond to conceptual design features. For example, the “imprint” conceptual design feature is transformed into the “completely adjacent” constraints, each of which is a co-surface relation between core insert, ejector pins, and the moulding after applying the shrinkage factors. The “shrinkage” constraint is applied on the interacting faces of the “imprint” feature. The

"guide" feature is transformed into relations between the ejector pins and the corresponding holes in the core insert. The "propel" feature is transformed into the constraint of contact area between the ejector pins and the moulding. As the final step of detailed design, the designer associates features created in the detailed design stage to those created in the conceptual design stage via a user interface as shown in Figure 7.14.

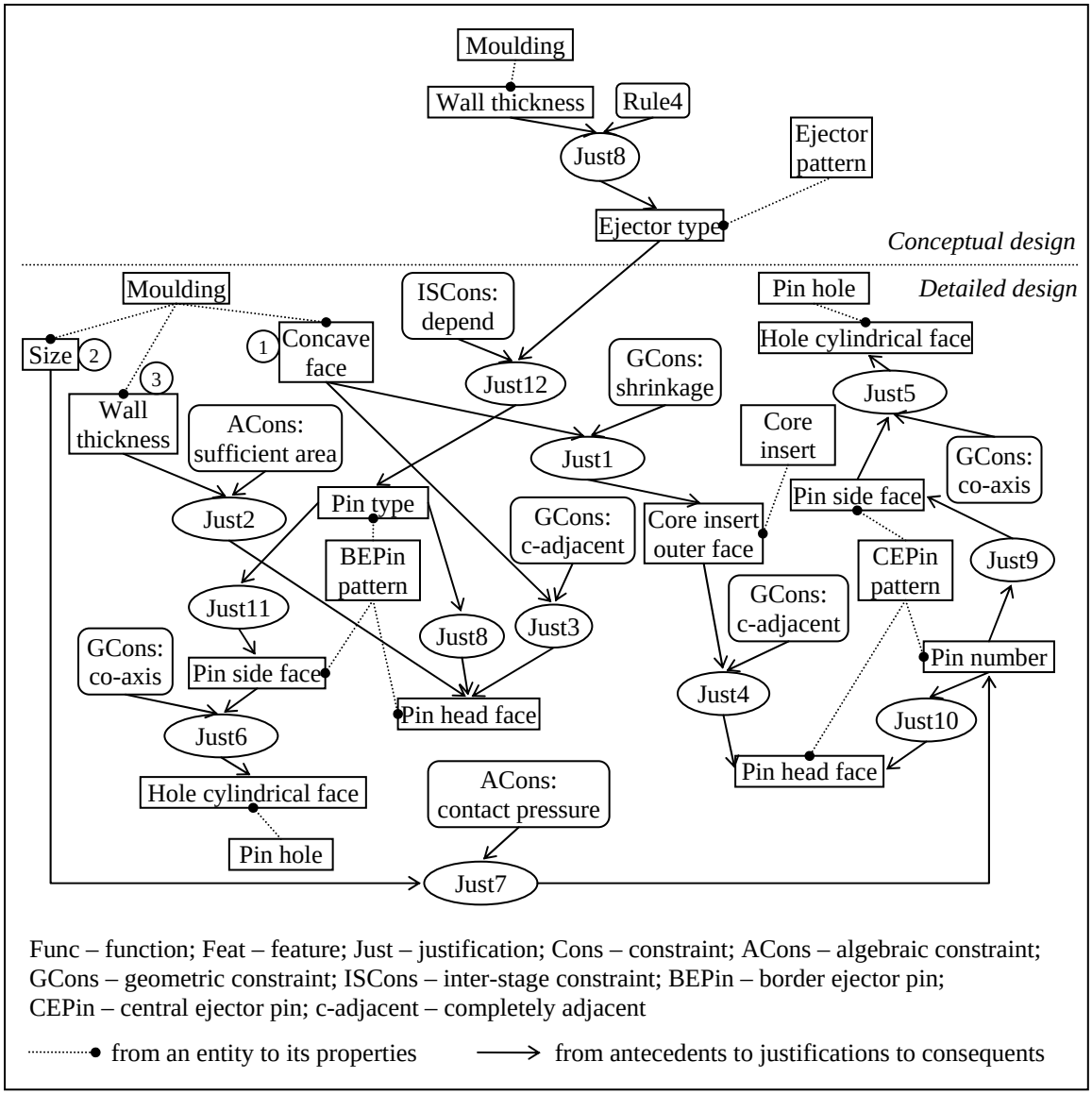


Figure 7.15 The JTMS dependency network for the ejection system in detailed design

The dependency network for the detailed design (recorded using JTMS) is established (Figure 7.15). Only one inter-stage association is shown in the figure. In the following sub-sections, how the association mechanism is used to find the affected entities (shown using bold and italic fonts in the figures) as well as how the validity of

the detailed design is checked, are illustrated with three modification scenarios. This mechanism is essential to maintain the consistency between the conceptual and the detailed design feature models.

Change Propagation and Consistency Evaluation

- Change 1 – increasing the radius of the concave face of the moulding

  When the spherical radius of the moulding face is changed, its corresponding JTMS node ("concave face" in Figure 7.16) is found in the JTMS dependency network. The modification is propagated (using the algorithm in Section 6.5) through a sequence of antecedents, justifications, and consequents.

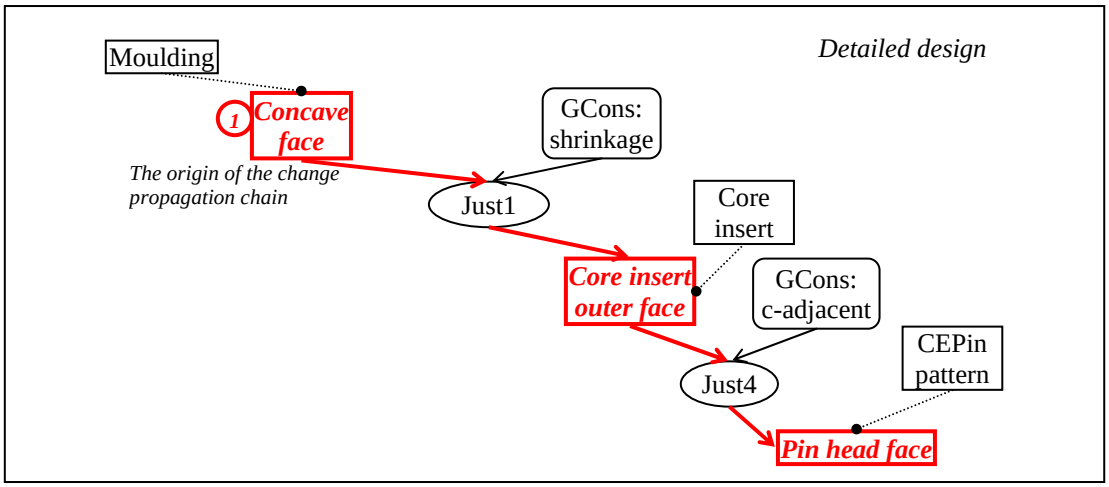


Figure 7.16 Changing the radius of the concave face of the moulding

As shown in Figure 7.16, which is an extract of Figure 7.15, the face of the core insert and the head faces of the central ejector pins are found to be inconsistent with the new moulding. This is signaled to the designer using the message box shown in Figure 7.17. The system solves the constraints, i.e. it changes the radius of the core insert and pin head faces automatically according to the new radius of the moulding face.

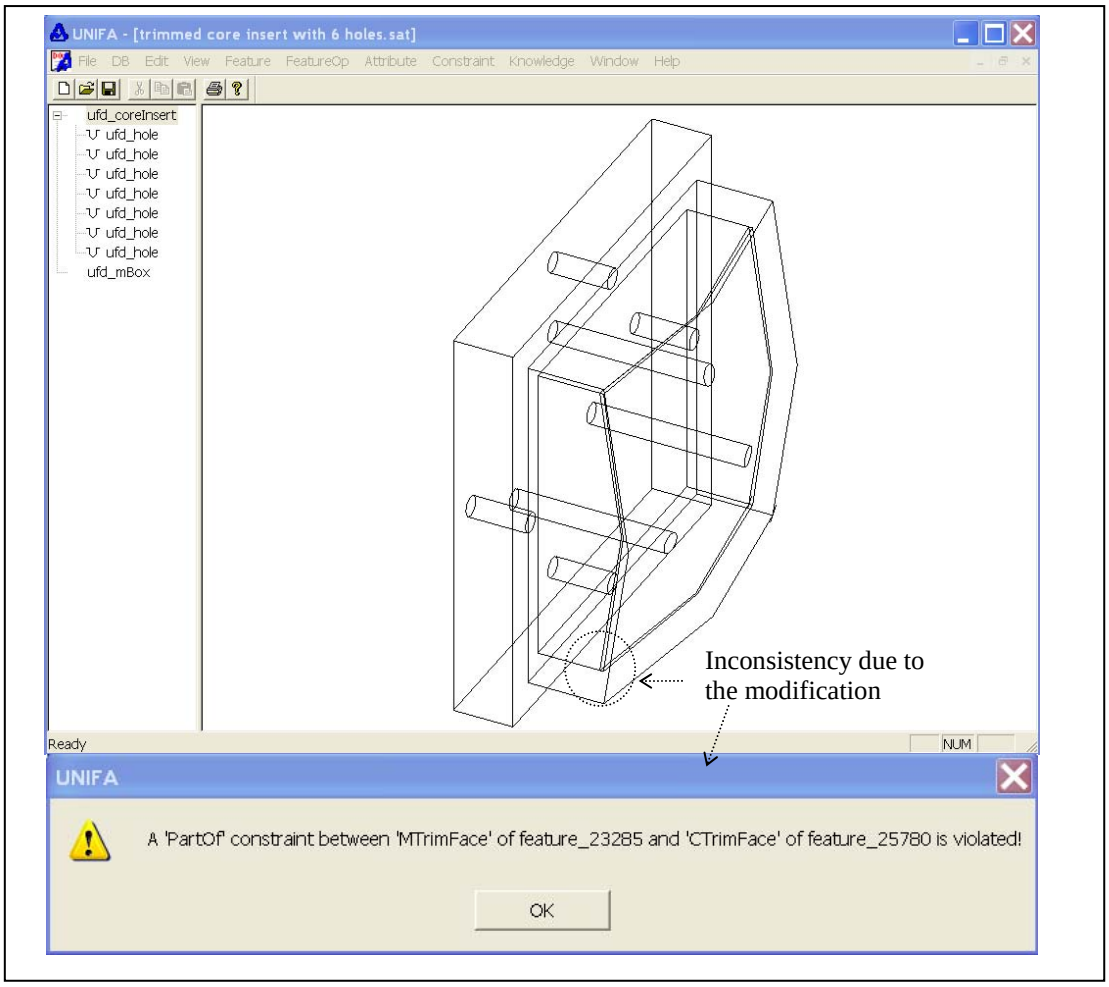


Figure 7.17 The core insert is inconsistent with the moulding due to the modification

- Change 2 – increasing the size of the moulding

  Similarly, when the size of the moulding is changed (here enlarged), its corresponding JTMS node is found in the JTMS dependency network. The modification is propagated (Figure 7.18). According to the algebraic constraints ("contact pressure") between the moulding size and the number of pins in the central pin pattern, the number of pins is insufficient for the enlarged moulding. Solving the constraints results in additional pins added in the central pin pattern. Following the dependency network, the system determines that increasing the number of pins implies two further modifications:

  (i) Because of the change to the pin positions and the "completely adjacent" constraints ("Just4"), the shapes of the pin head faces need to be changed.

(ii) Because the holes in the core insert have co-axis constraints ("Just5") with the pins, the positions of holes must be changed accordingly. In addition, for the newly added pins, as they correspond to one of the interacting faces of the "guide" conceptual design feature, the integrity of the "guide" feature requires new holes in the core insert according to the positions and size of the new pins.

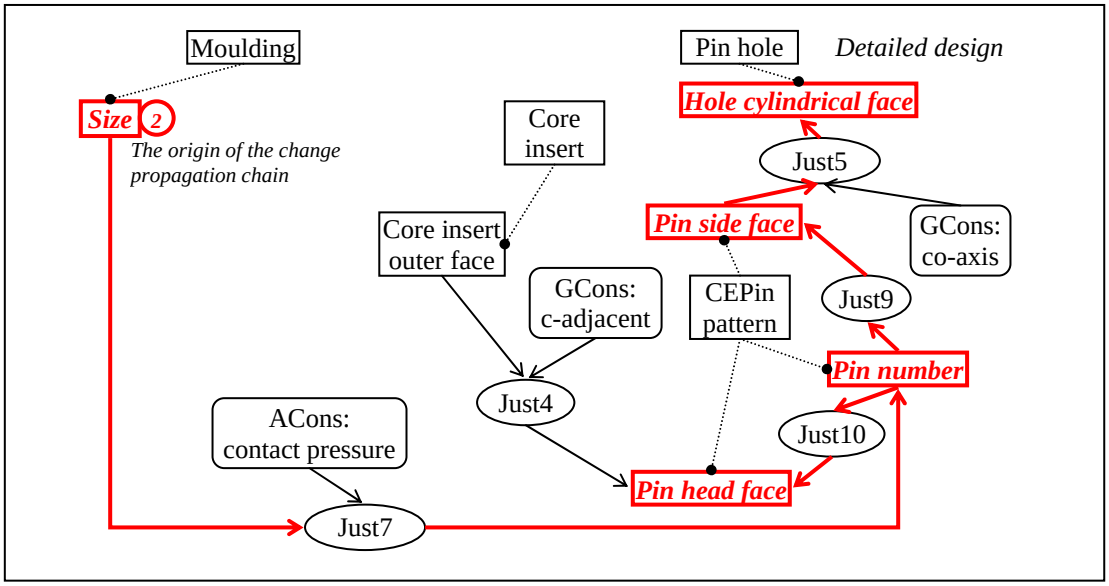


Figure 7.18 Changing the size of the moulding

- Change 3 – decreasing the wall thickness of the moulding

When the wall thickness of the moulding is changed (here reduced), the modification is propagated starting from the corresponding JTMS node in the dependency network (Figure 7.19). In this case study, the wall thickness is reduced such that the algebraic constraint of "sufficient contact area" between the walls of the moulding and the pin head faces is violated (Figure 7.20). In general, several counteractions to increase the contact area between the moulding and the ejector pins exist: changing the diameters, positions, or the type of pins (Figure 7.21).

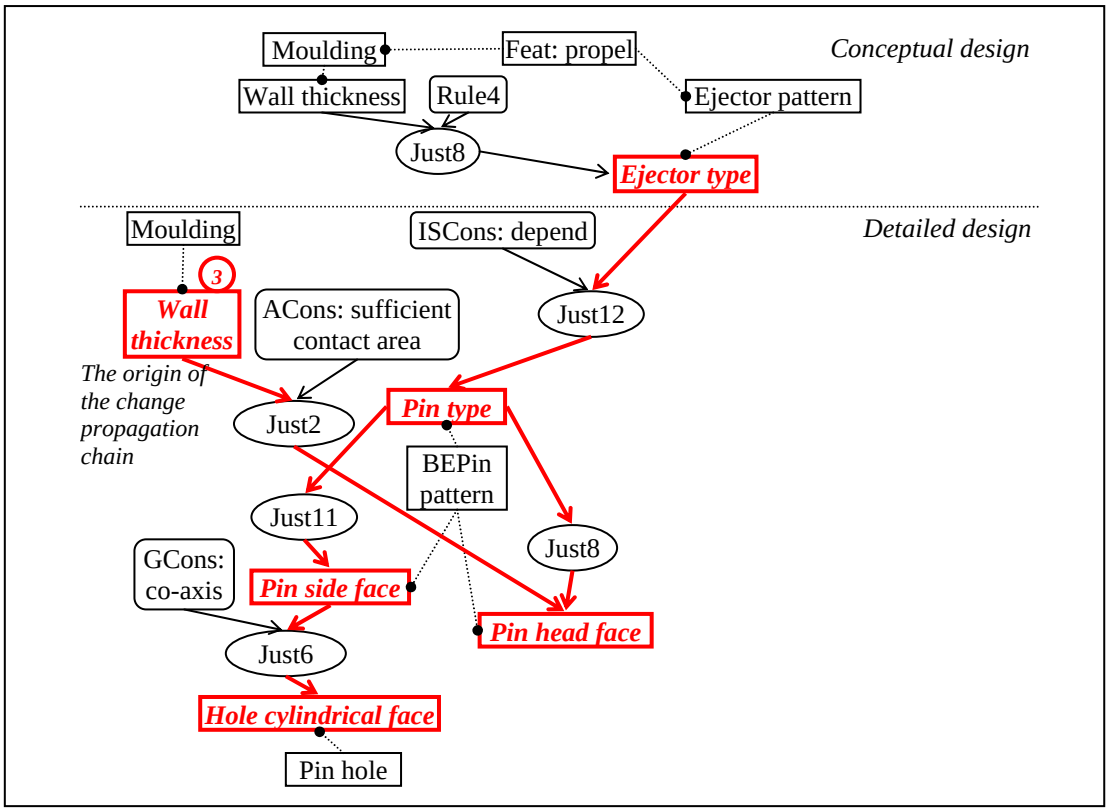


Figure 7.19 Changing the wall thickness of the moulding

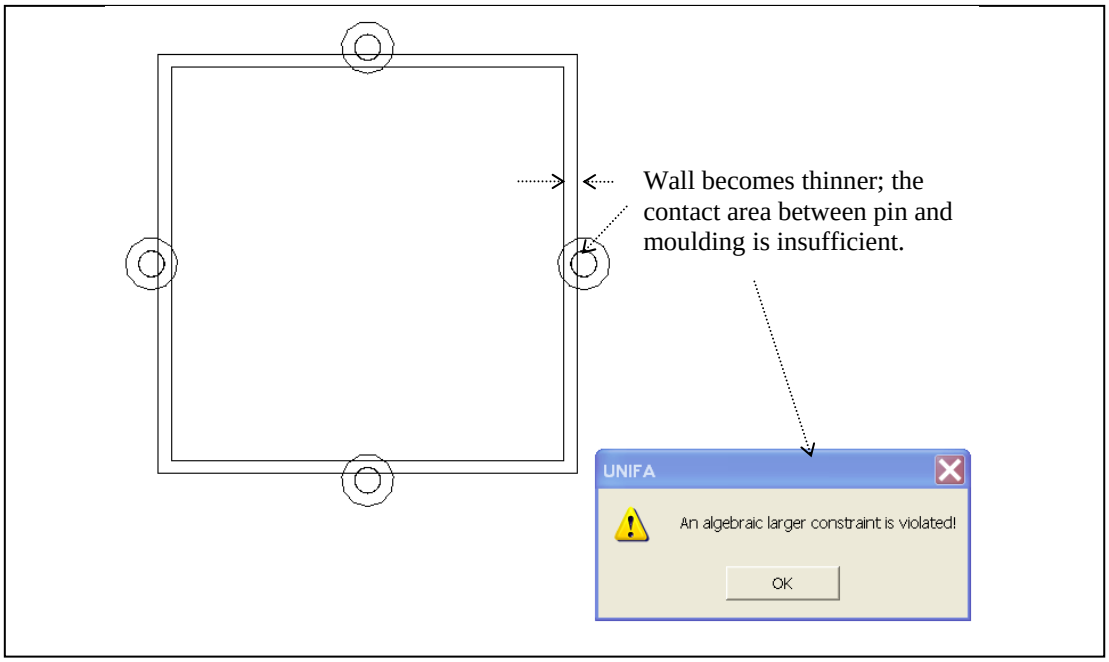


Figure 7.20 New wall thickness violates the “sufficient contact area” constraint

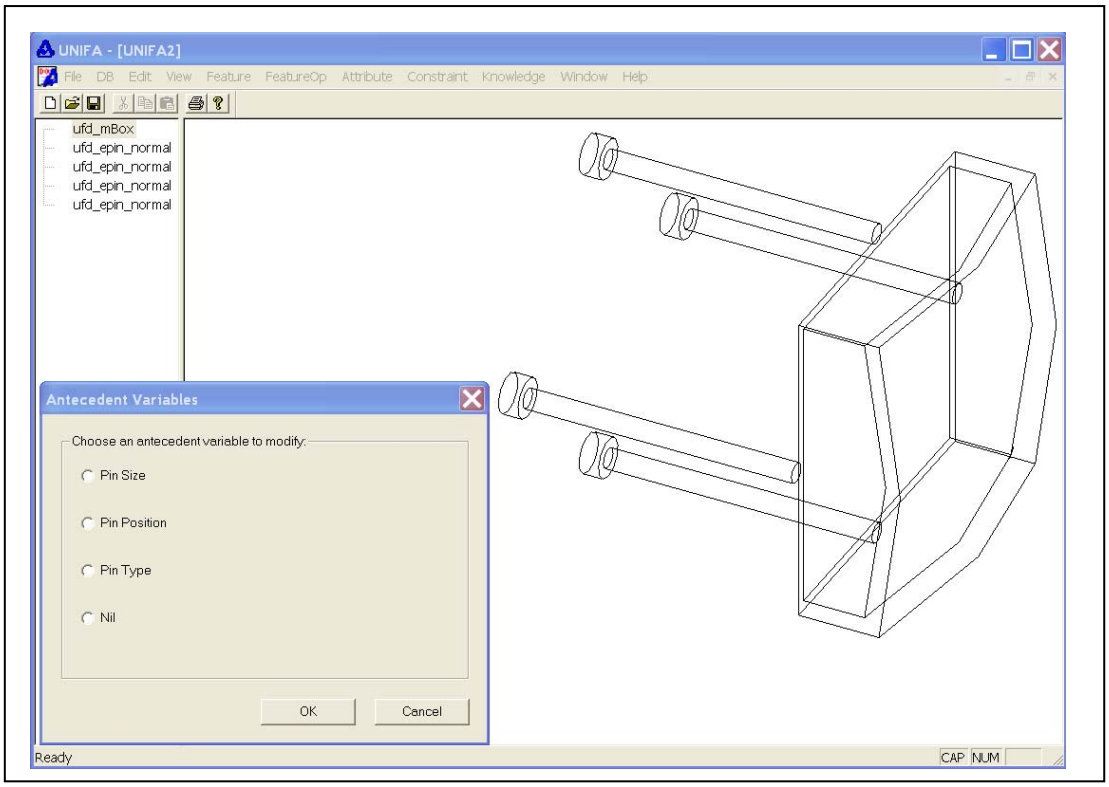
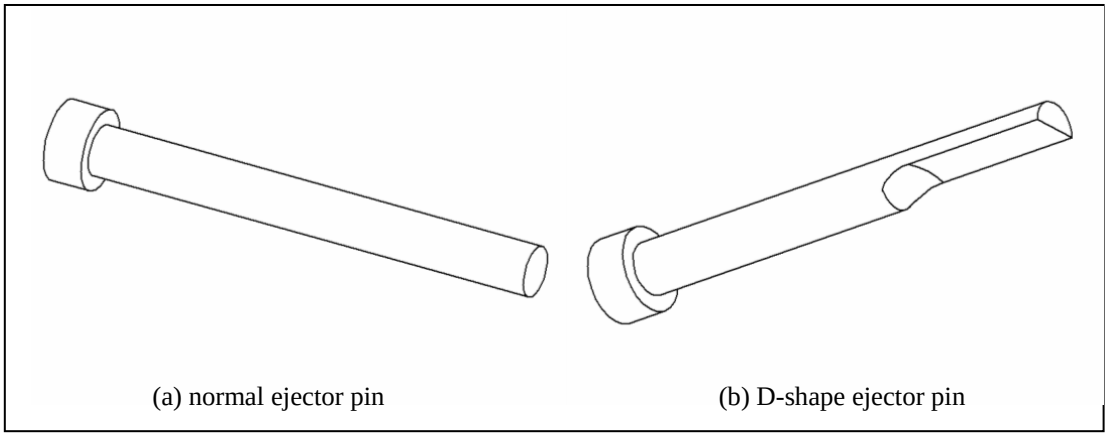


Figure 7.21 Choices for re-satisfying the contact area constraint

However, in this case, the first two choices are infeasible due to the pins' interference with the core insert. Changing the pin type is a feasible solution. Compared with the normal ejector pins, commonly used D-shape ejector pins are chosen here because they can provide bigger contact area (with the same pin diameter) to eject thin-wall moulding (Figure 7.22).



(a) normal ejector pin (b) D-shape ejector pin

Figure 7.22 Normal and D-shape ejector pins

After the designer changes the pin type from normal to D-shape pin, the system checks the database and finds the modification ("pin type") associated to an entity ("ejector type") in the conceptual design. This modification is transferred into the conceptual design for validation.

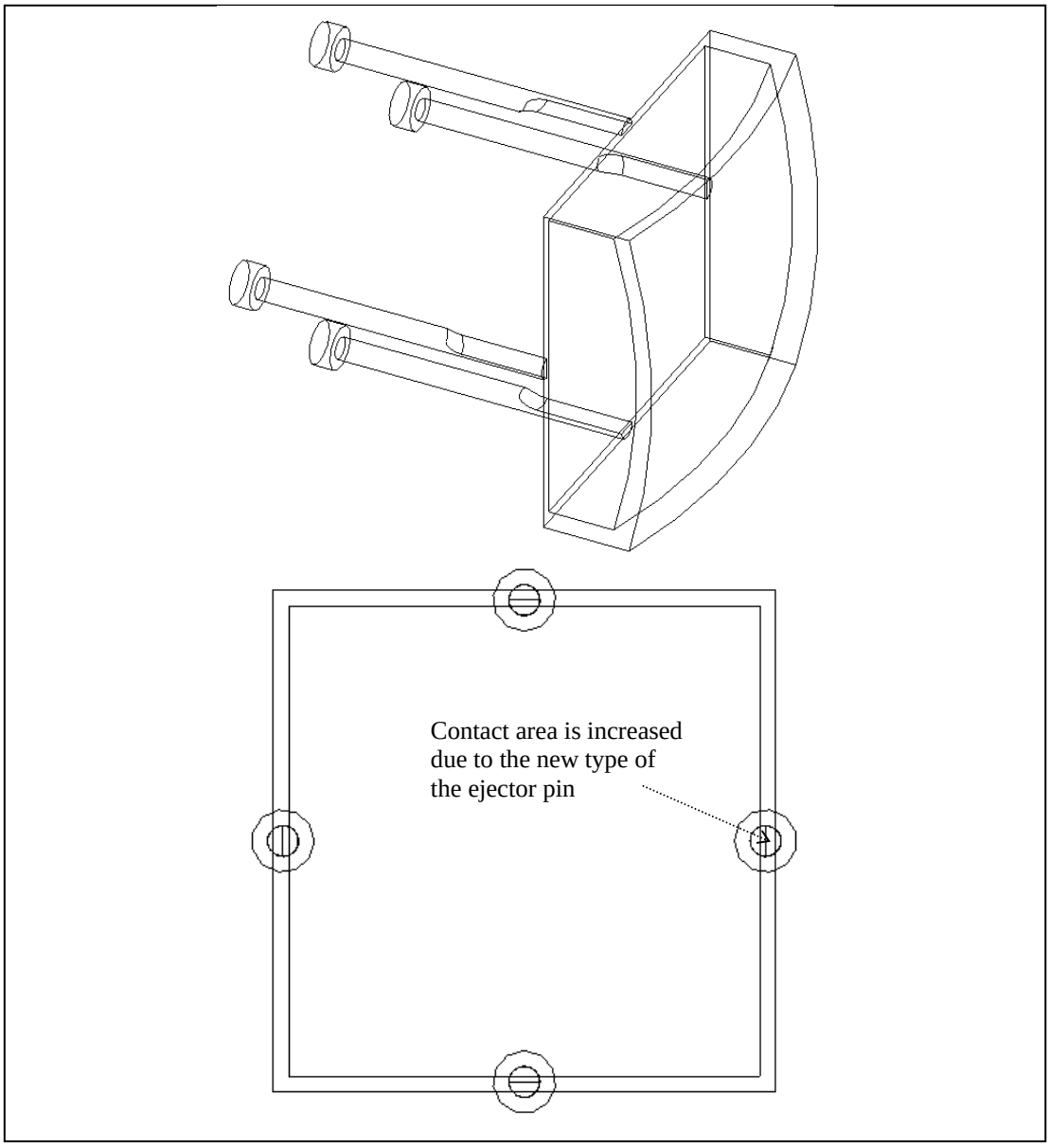


Figure 7.23 The pin type is changed to "D-shape" to increase the contact area

The JTMS dependency network of the conceptual design is consulted and an "Ejector Type Rule" (rule4) is found to justify the "ejector type" entity. Since "D-shape ejector pin" is one of the alternative consequent options of the rule, the modification is permitted (Figure 7.23). As for the detailed design application,

changes to the pin head faces as well as the guiding faces (pin holes) in the core insert are effected according to the new pin type.

7.2.2 Association between Detailed Design and Process Planning Stages – Case 2

The second case illustrates the association and change propagation within and between the detailed design and process planning stages.

Generating the Detailed Design of the Part

The original part is a block with a central pocket, a through hole, and two side blind holes as shown in Figure 7.24. The numbers in the figure are the face identifiers.

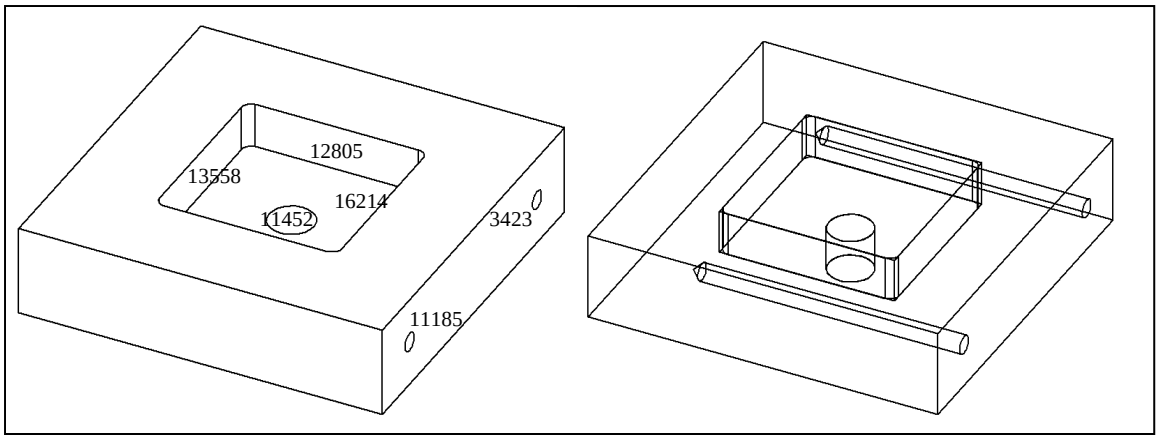


Figure 7.24 The original part and its cellular model

Design specifications, such as the dimension, tolerance scheme, and surface roughness, can be added or edited as shown in Figure 7.25.

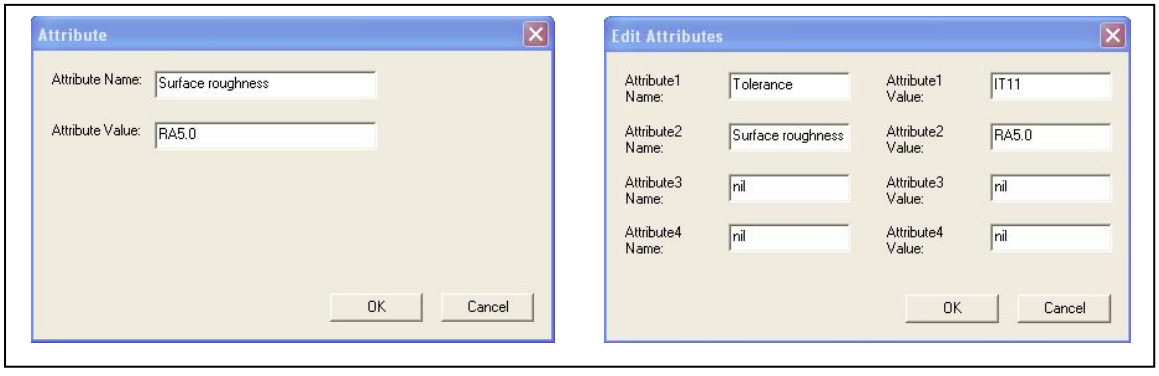


Figure 7.25 Adding (or editing) attributes of a detailed design

Process Planning

The process planning application reads and displays the data of a detailed design, including features and other specifications, through enquiring the database. The used data includes the machined faces, face types, dimensions, tolerances, and surface roughness (Figure 7.26).

- Step-1: Machining methods and route for each machined face

  With the predefined rules for machining method selection [196], which are loaded from the database, the expert system generates the machining route for each machined face according to its tolerance and surface roughness specifications (Figure 7.26). The generated machining route objects are associated to the corresponding machined faces. The dependency associations are recorded in the JTMS dependency network.

Face identifier	Face type	Face size	Tolerance	Surface roughness	Machining route
face_11185	I_CYL	4	IT11	RA5.0	drilling
face_11452	I_CYL	10	IT11	RA5.0	drilling
face_12805	PLANE	36*0*10	IT11	RA5.0	rough_milling
face_13558	PLANE	0*36*10	IT11	RA5.0	rough_milling

* I_CYL represents internal cylindrical face.

Figure 7.26 Machining routes for some machined faces

- Step-2: Machine tool and cutter for each machined face

  According to the generated machining routes, the system selects machine tools and cutters (type, section, and orientation) for each machined face (Figure 7.27).

Face identifier	Machining route	Machine tools	Cutter	Cutter section	Cutter orientation
face_11185	drilling	DM/HMM/VMM/HMC/VMC	Drill/End mill	Nil/End section	+x
face_11452	drilling	DM/HMM/VMM/HMC/VMC	Drill/End mill	Nil/End section	+z/-z
face_12805	rough_milling	HMM/VMM/HMC/VMC	Side/End mill	Side section	+z
face_13558	rough_milling	HMM/VMM/HMC/VMC	Side/End mill	Side section	+z

DM: drilling machine HMM: horizontal milling machine VMM: vertical milling machine
HMC: horizontal machining center VMC: vertical machining center

Figure 7.27 Machine tools and cutters for some machined faces

- Step-3: Operation elements

  Operation elements are generated by grouping faces, which can be machined with the same combination of machine tool, cutter, cutter orientation, locating and clamping methods (Figure 7.28).

Operation element	Member face identifier	Cutter, cutter section, and cutter orientation
OE_10186	18434	E_S_-x/S_S_-x/E_S_-z/S_S_-z/ E_E_+y/E_S_+x/S_S_+x/E_S_+z/S_S_+z
OE_19615	11452	E_E_-z/D_N_-z/E_E_+z/D_N_+z
OE_24052	3423	E_E_+x/D_N_+x
OE_29685	16214/9973/4441/9597/13558/ 9816/12805/9898/19371	E_E_+z

Figure 7.28 Some operation elements (following notations in Section 5.2.2)

- Step-4: Locating and clamping methods for each operation element

  As described in Section 5.2.2, the system then determines the locating faces for each machined face according to the design dimension scheme. In addition, a pair of parallel faces on the workpiece, defined as a clamping feature, is identified as the clamping faces for each operation element and associated to the operation element. The geometry of a clamping feature is represented as a pair of face cells and inserted into the process planning cellular model as well as the unified cellular model.

- Step-5: Machining sequence

  With the predefined machining sequence rules, the system orders the operation elements (Figure 7.29). The process planner can adjust the sequence according to his desire.

- Step-6: Setups

  Machining operation elements, which can be carried out using the same machine tool, cutter orientation, the workpiece locating and clamping methods, are grouped into a single setup. The sequence of setups is determined by the sequence of their member operation elements. The example part is machined using four setups (refer to Figure 5.11).

- Step-7: Process planning features

  The shape of a process planning feature is determined by its belonging operation element, i.e. the machining method, cutter (type, section, and orientation), and

machining sequence. Figure 7.29 illustrates the generation process of process planning features.

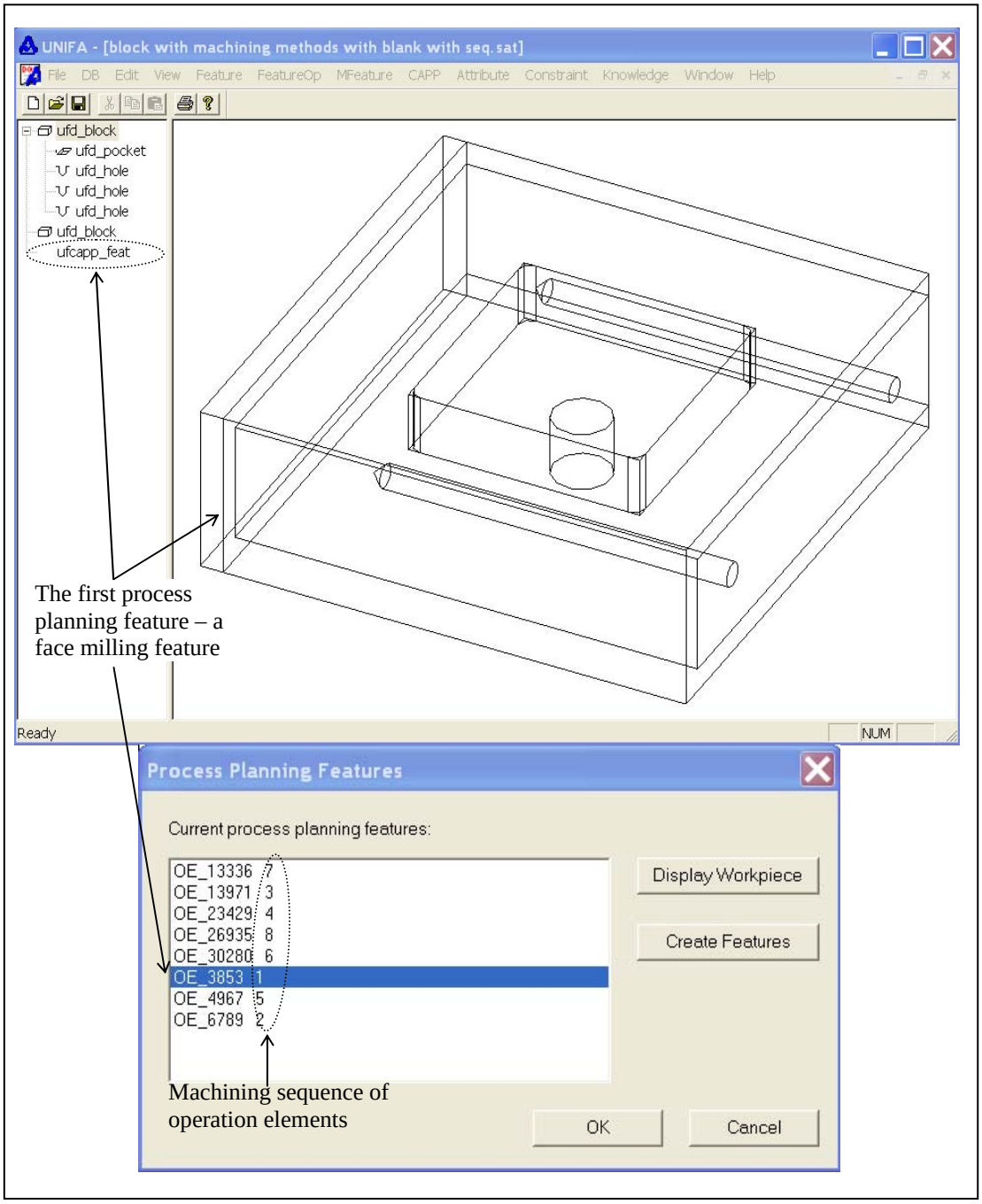


Figure 7.29 The first process planning feature – a face milling feature

Figure 7.30 shows the final process planning feature model (consisting of eight process planning features) of the original part. Each process planning feature is associated with the corresponding operation element. If a process planning entity

(e.g. the machining route object) is generated directly from the design specification, the inter-stage associations are established and stored in the database.

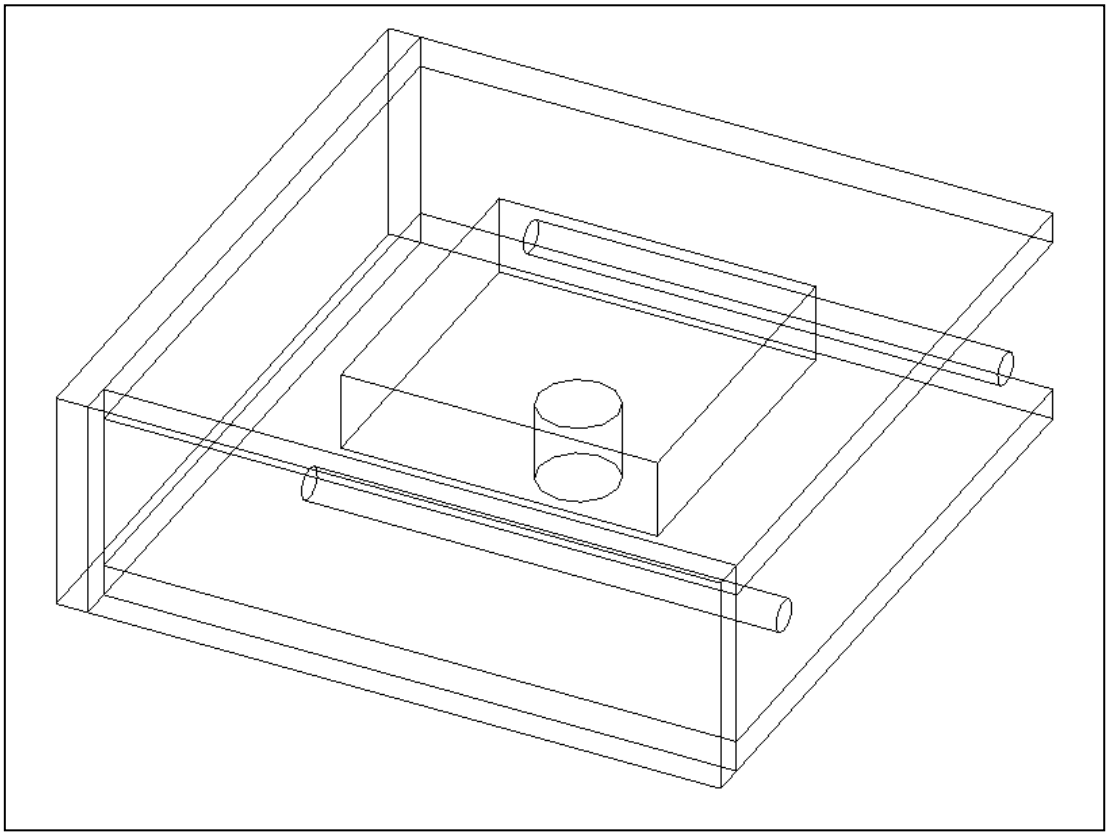
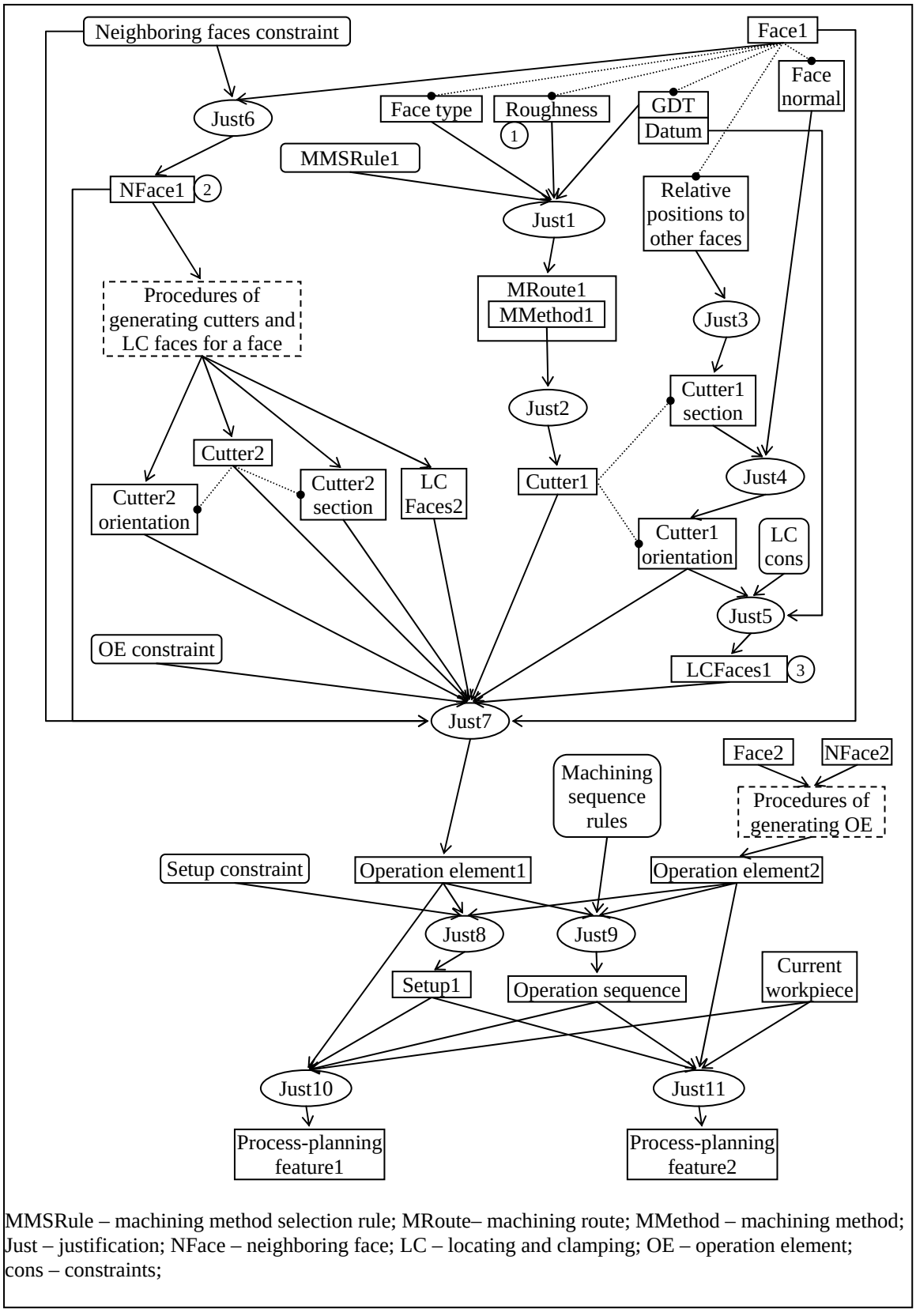


Figure 7.30 Process planning feature description of the original part

After process planning, a dependency network (recorded using JTMS) is established (Figure 7.31). To avoid redundancy, some entities are not shown in the figure (represented as two dashed squares). When making product modifications, the system refers to this dependency network for the affected entities as well as the validity checking. In the following sections, two modifications are used to illustrate how the association mechanism detects the affected entities, checks the validity of process plans, and maintains the consistency between the detailed design and the process planning feature models.



MMSRule – machining method selection rule; MRoute– machining route; MMethod – machining method; Just – justification; NFace – neighboring face; LC – locating and clamping; OE – operation element; cons – constraints;

Figure 7.31 The JTMS dependency network for the process planning case

Change Propagation and Consistency Evaluation

- Change 1 – changing the surface roughness of the central through hole

When the surface roughness specification of the central through hole is changed in the detailed design, the modification is published and checked into the database. Once the process planning application checks the database and gets the modification, the process planning JTMS dependency network is searched for the corresponding JTMS node, "Roughness" in Figure 7.32. It is found that a machining route object depends on the modified surface roughness. In addition, the machining route object is derived based on the justification of a machining method selection rule ("MMSRule1" in Figure 7.31).

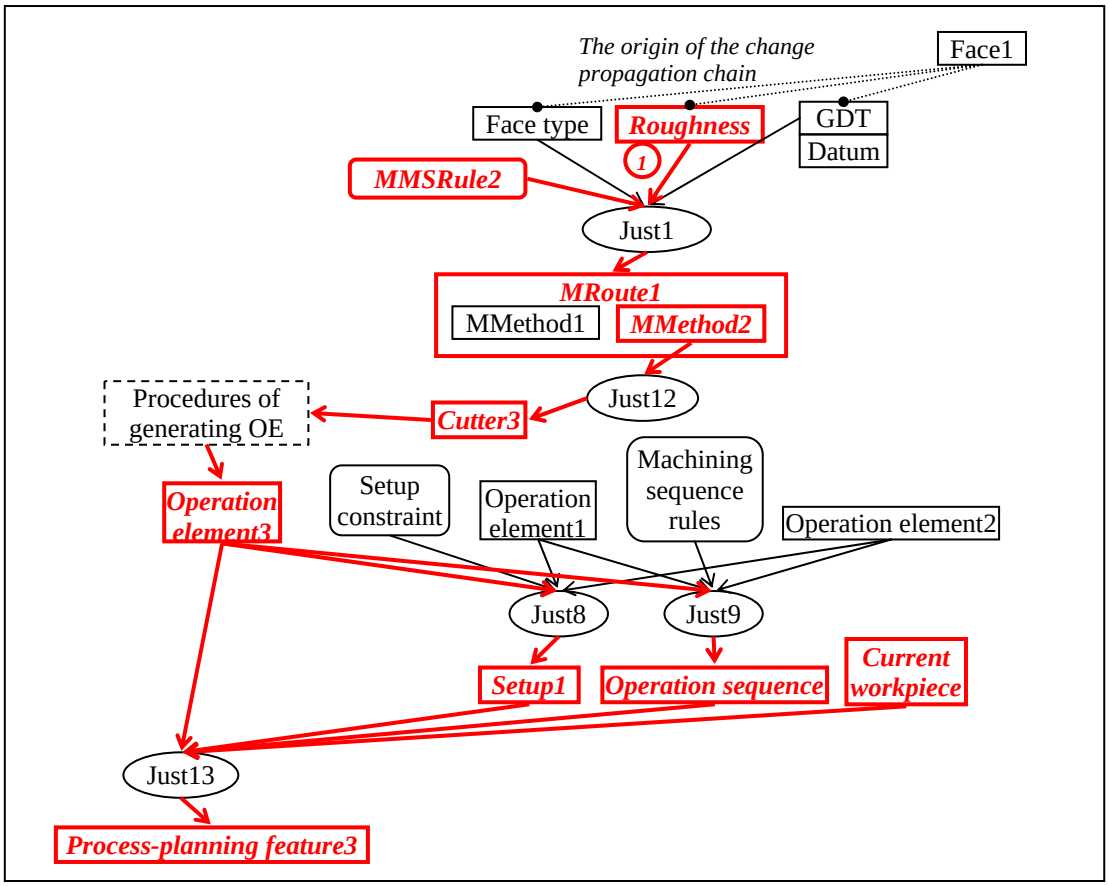


Figure 7.32 Changing the surface roughness specification of the central through hole

Re-running the expert system revokes the existing rule and fires another selection rule according to the new surface roughness of the face. As the consequence, the affected machining route object is modified. Because the surface roughness is changed from RA5.0 to RA2.5, the machining route is changed from "drilling" to

"drilling and semi-finish boring" accordingly. As the modification is propagated, it is found that the previously assigned cutter (a drill) is not enough for the new machining route. A boring tool is hence added. At the same time, a new boring operation element is generated within the existing setup 1, i.e. using the same machine tool, cutter orientation, workpiece locating and clamping methods. The new operation element is ordered right after the "drilling" operation element. The corresponding process planning feature, a boring feature is generated. The machining allowance of the "drilling" operation element is adjusted as well to leave sufficient allowance for the boring operation (Figure 7.33).

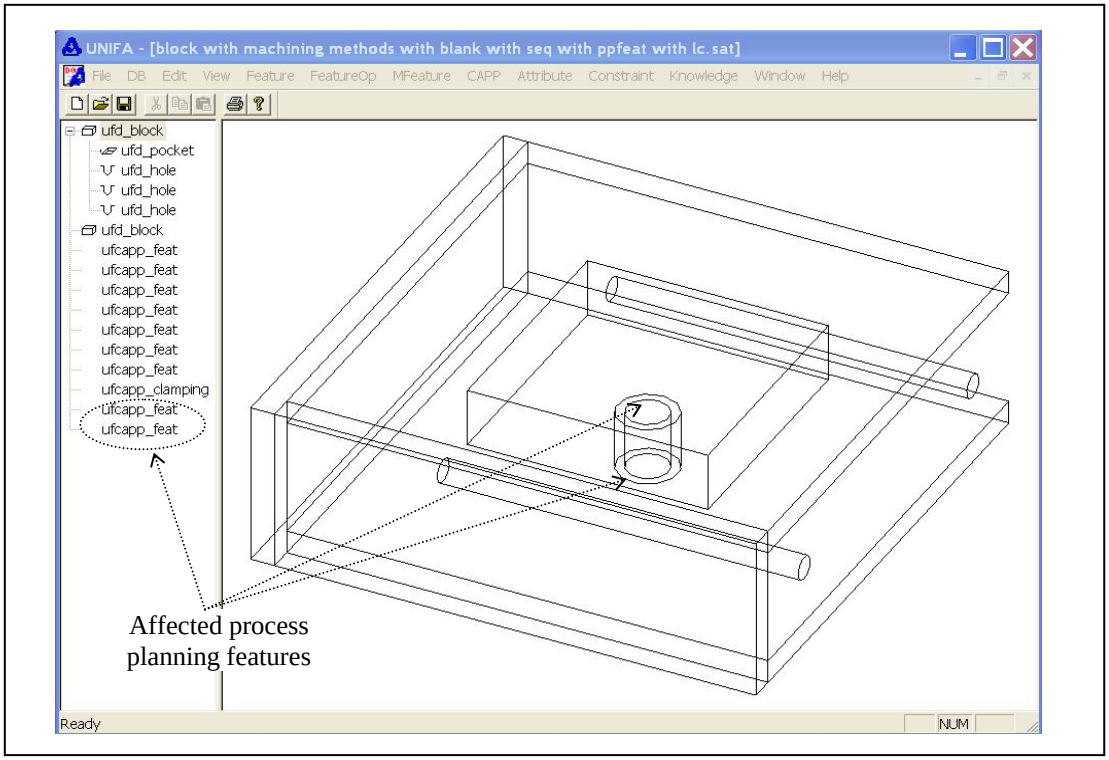


Figure 7.33 A new process planning feature is generated due to the design change

- Change 2 – adding a new feature (a side pocket) to the detailed design

  After the designer adds a new feature to be machined, a side pocket, to the detailed design as shown in Figure 7.34, the new feature is stored in the database. Its cell is also inserted into the unified cellular model.

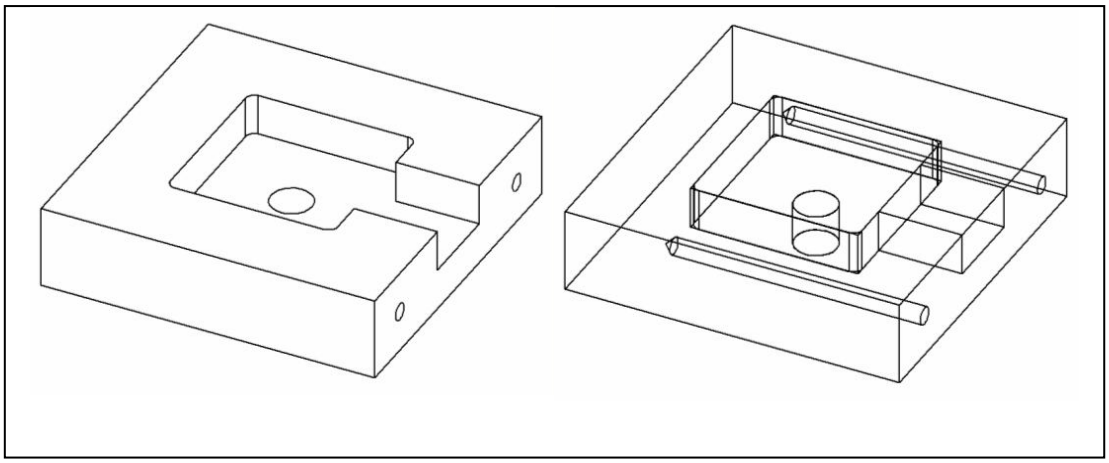


Figure 7.34 The modified part (a new side pocket) and its cellular model

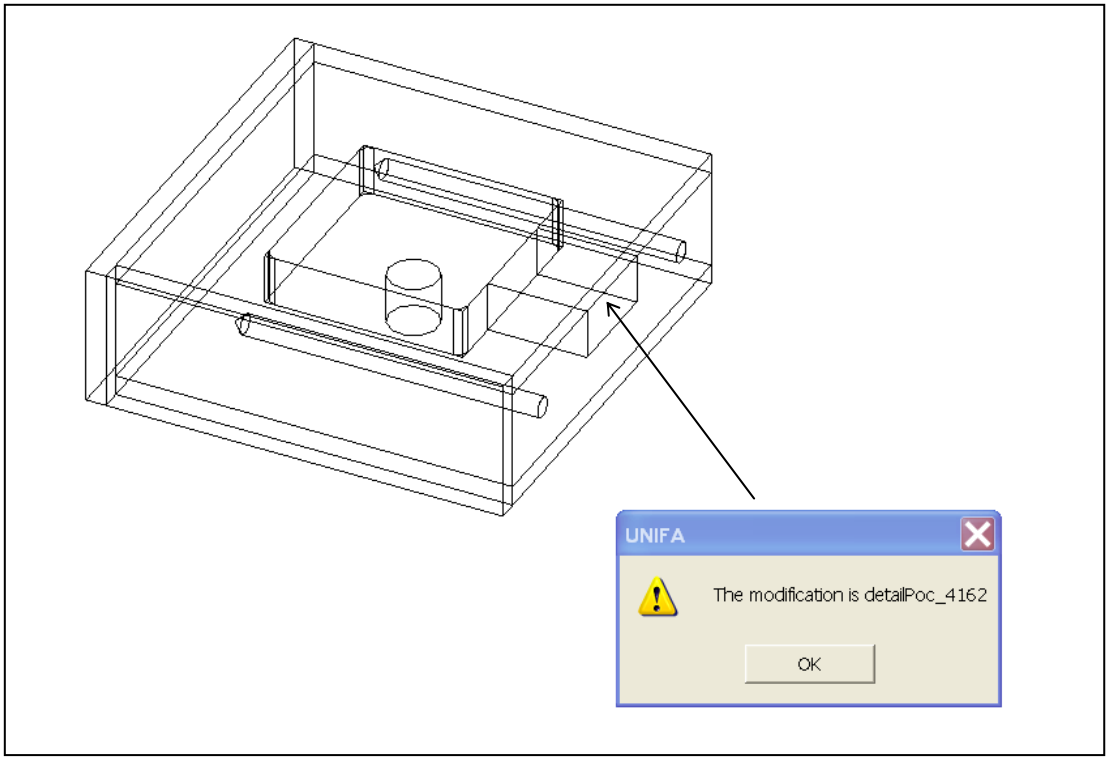


Figure 7.35 The process planning application gets the design modification

The unified cellular model automatically discovers that this new cell does not have corresponding process planning features in its owning feature list. According to the integrity criteria that “each detailed design feature to be machined is linked to process planning features via valid constraint-based associations” (Section 6.4), the process plan becomes incomplete due to the modification. The unified cellular model notifies the process planning application about the modification (Figure 7.35).

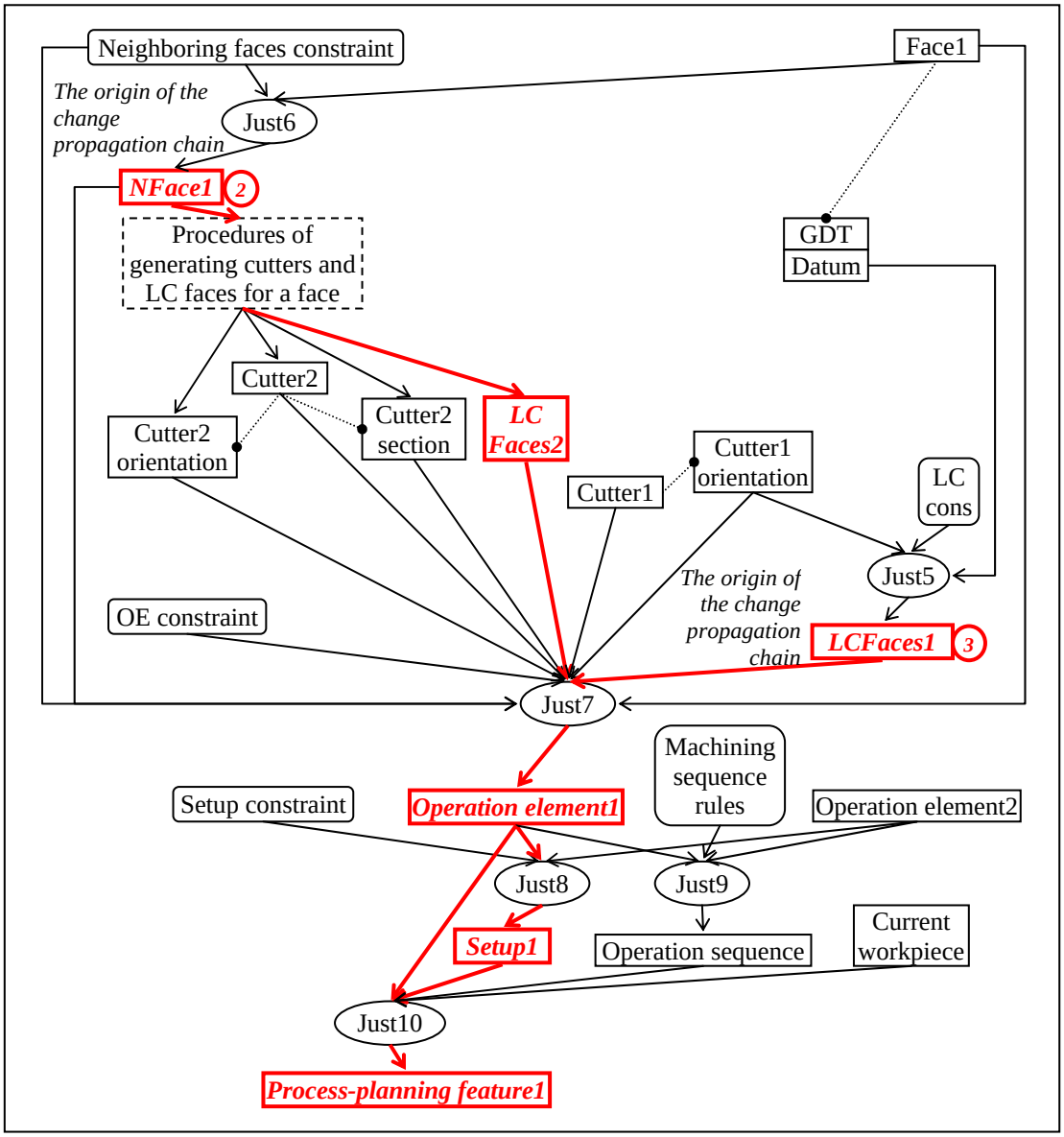


Figure 7.36 Adding a new feature to the detailed design

On the process planning side, after checking its JTMS dependency network, it is found that this design modification invokes two corresponding modifications to the original process plan (Figure 7.36):

(i) The characteristics of the new design feature are evaluated first. The machining routes, machine tools, cutters, locating and clamping methods for new faces are then generated accordingly. It is found that three faces (the other three are virtual faces) of the new feature can be grouped into an existing operation element (the milling operation for the previous central pocket). The operation element 1 and

its corresponding process planning feature are modified to the shape as shown in Figure 7.37.

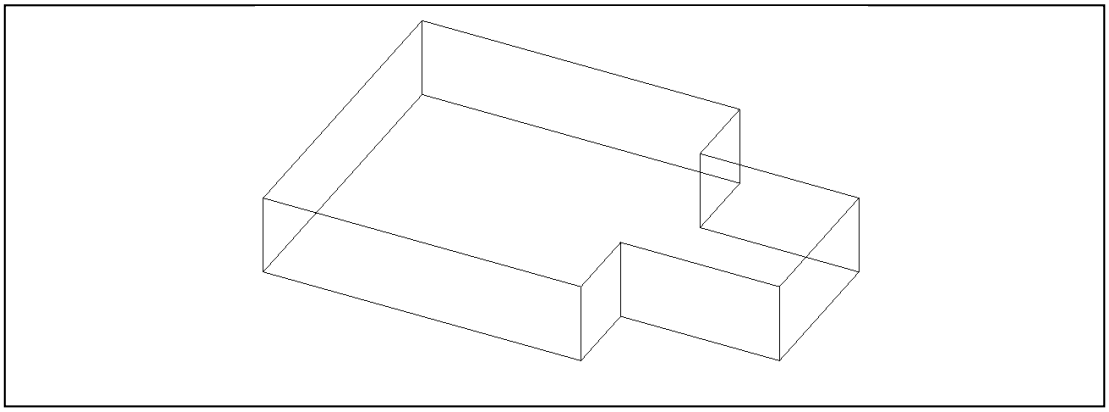


Figure 7.37 A process planning feature is modified to accommodate the new design feature

(ii) Because the geometries of clamping features are also represented in the unified cellular model as face cells (Section 4.3.4), it can be detected that the modified milling feature affects the clamping option from the opening side of the pocket (because they share the same face cell, Section 4.3.5). It is then found that a non-interference constraint ("LC cons" in Figure 7.36, see also Section 4.3.5) is no longer satisfied. The system sends a warning message to the process planner and requires changing to the other clamping option as shown in Figure 7.38.

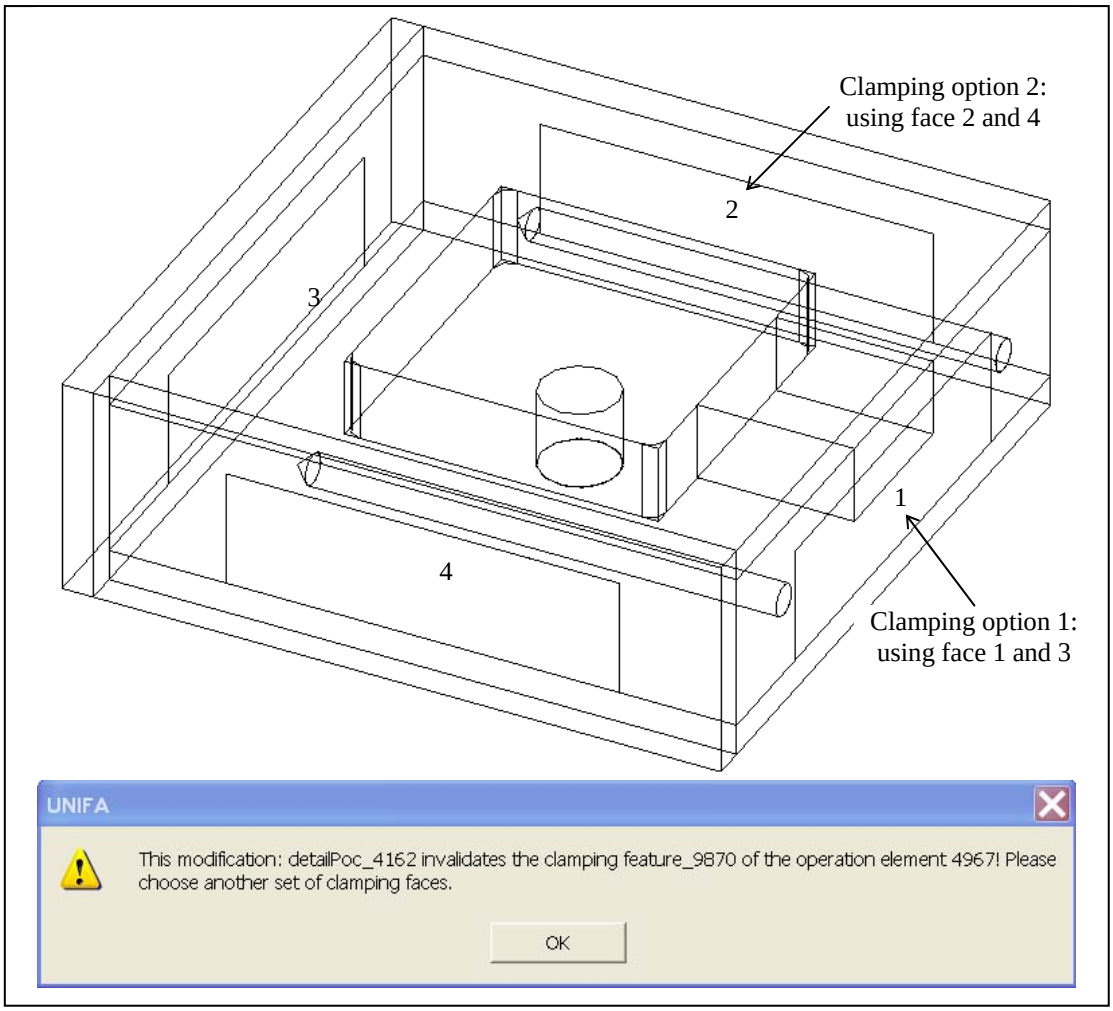


Figure 7.38 The process planning application identifies that a clamping method is invalid

7.2.3 Association between Detailed Design and Process Planning Stages – Case 3

A simple part for connection (Figure 7.39) is used as an example to further illustrate the associations between the detailed design and the process planning stages. The differences between this and the previous case are:

- The design is created using both additive and subtractive form features. The process planning application can handle this situation.
- It uses process planning rules to verify design modifications.
- Using the unified feature and the database to share data between stages is illustrated.

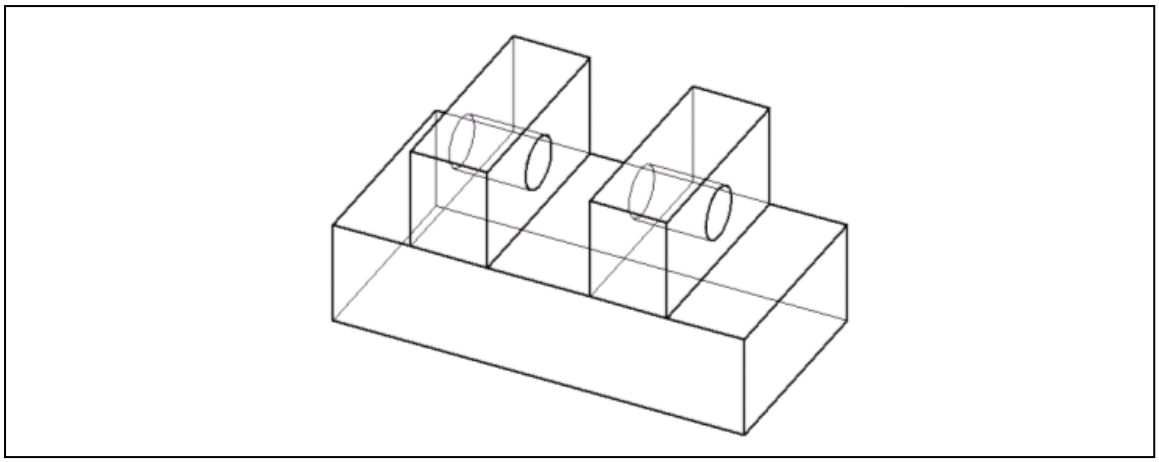


Figure 7.39 The example part

The Detailed Design Stage

The example part has five features, which are one base block, two boss and two hole features (Figure 7.40).

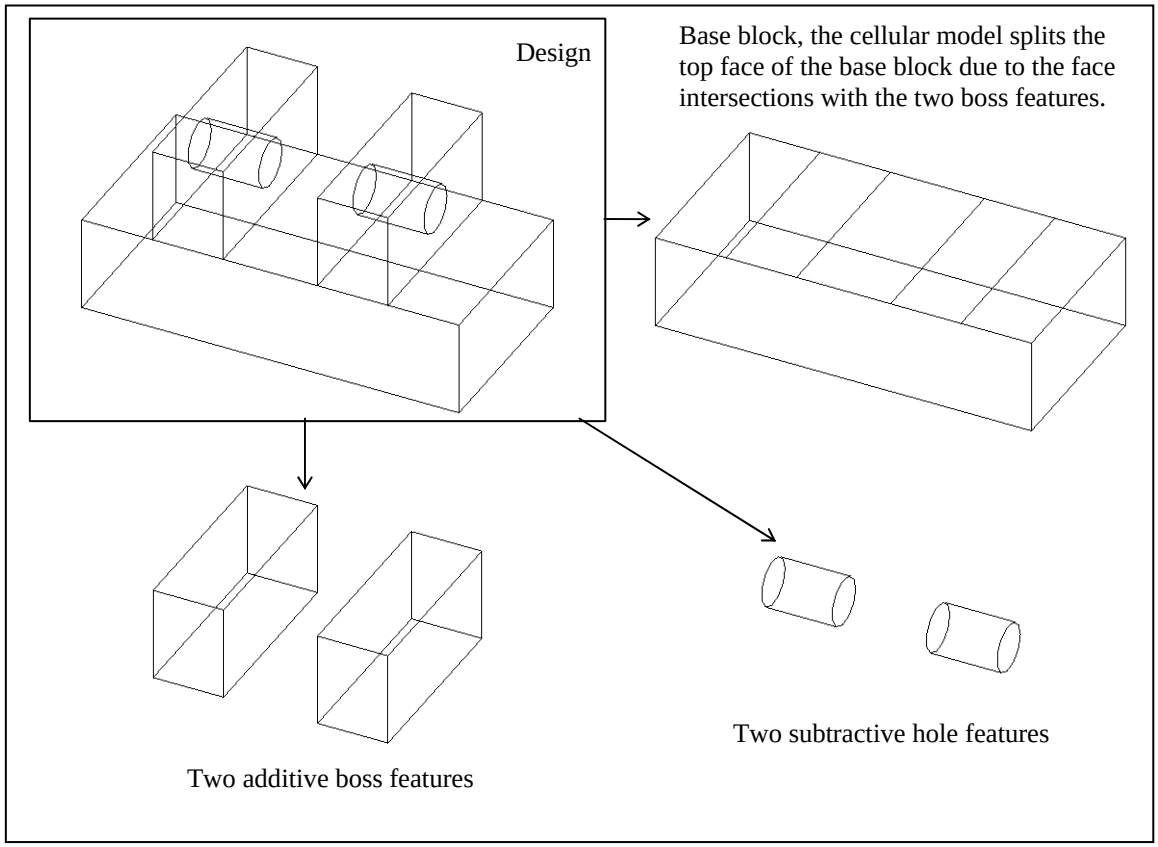


Figure 7.40 The detailed design of the part

Dimensional tolerances and surface finishes are specified as feature attributes. No concentricity tolerance is specified between two holes. During the design process:

- The dependency relations, such as the holes depending on the respective boss features as well as the corresponding reference faces and the relative positions, are specified as constraints and recorded using JTMS.
- The feature parameters, attributes, and constraints are stored into the database.
- In the unified cellular model, each solid cell and face cell are tagged with the identifiers of their belonging features.

The Process Planning Stage

In this stage, the process planner first re-creates the part design by querying the database. The design specifications, such as the tolerances and surface finishes, are also retrieved. The corresponding facts are generated and inserted into the process planning knowledge base. Similar to the steps described in the previous section, the process planning procedures are executed, which include creating a blank, identifying the machined faces, generating machining routes and finding cutters for each machined face, clustering machined faces as operation elements, finding clamping faces for each operation elements, determining machining sequence and setups. Figure 7.41 shows the part design with a generated blank and the identified machined faces. When generating machining routes for each machined face, the associations between the design features and the generated machining route objects are also automatically generated, recorded by JTMS, and stored in the database.

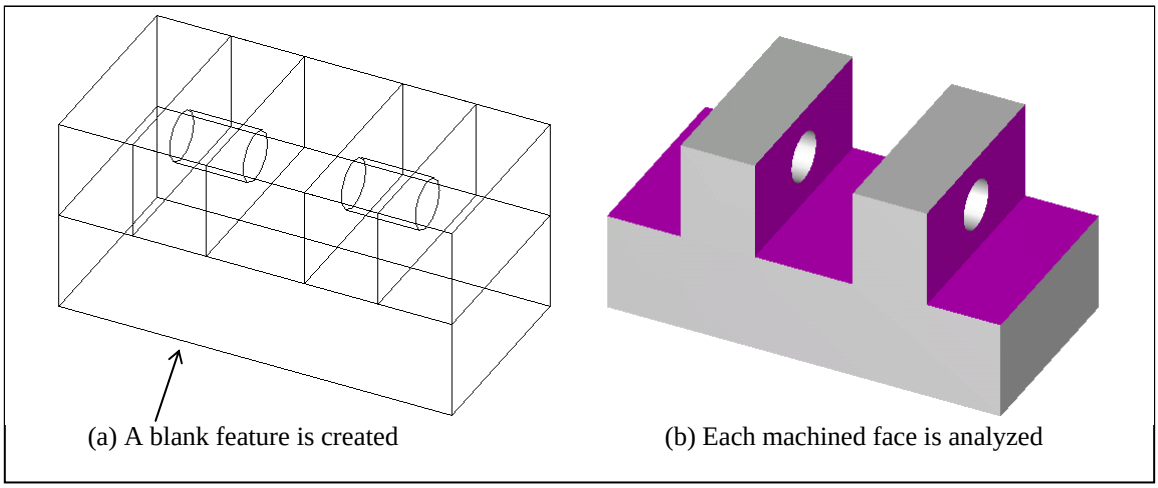


Figure 7.41 The design is re-created, then process planning procedures are executed

Finally, six process planning features, which include three milling features, two drilling features, and a blank feature, are created (Figure 7.42).

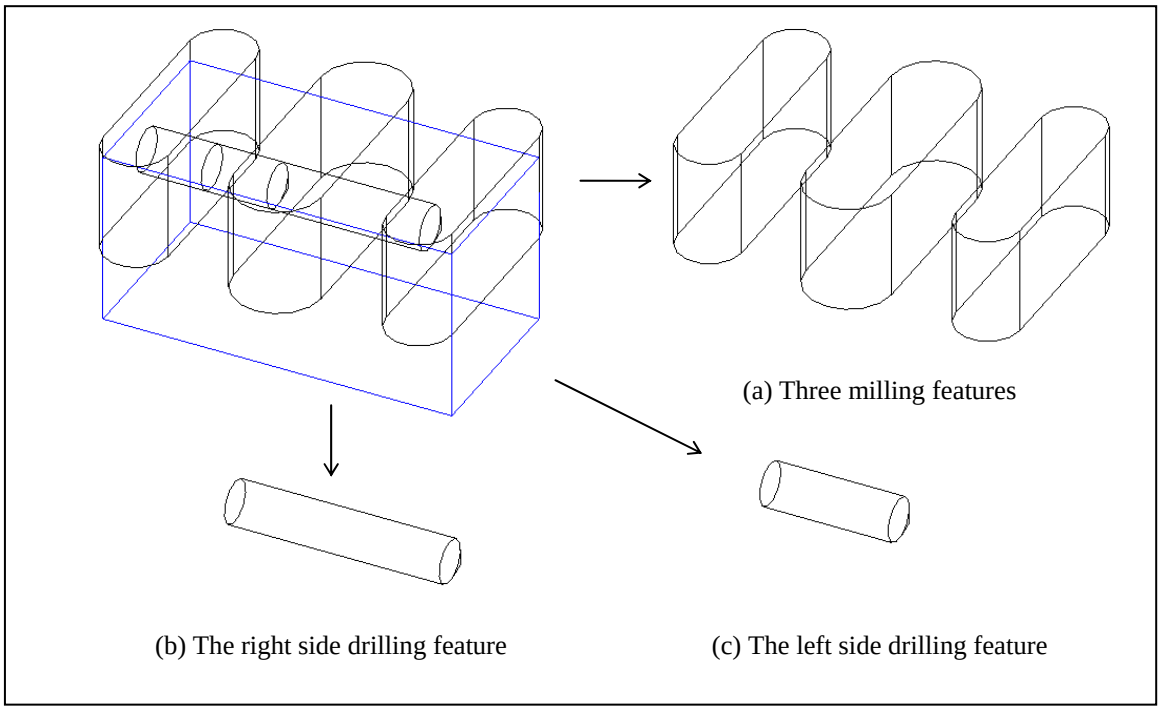


Figure 7.42 Process planning features

Three predefined process planning rules are fired:

- No interference between process planning features and the final part. This is a hard constraint.
- To minimize the numbers of setups, process planning features should be machined from the same direction if possible. This is the preference of process planners.
- The length/diameter ratio of twist drilling features cannot exceed a predefined value. This is a hard constraint.

Firing the third rule generates an algebraic constraint between the length and the diameter of the drilling features. These rules and constraints are associated with the respective features (Figure 7.43). The associations are recorded using JTMS and also stored in the database.

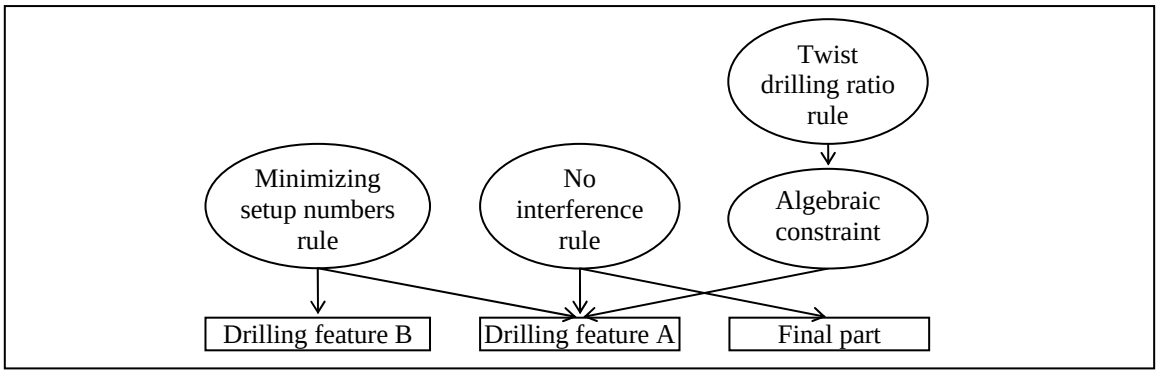


Figure 7.43 Associations between process planning features, rules, and constraints

Till now, the process planning is finished. The central database records two types of inter-stage associations.

- Geometry-level associations recorded by the unified cellular model. Figure 7.44 shows the cellular model after the process planning features have been created.
- Feature-level non-geometric associations. For example, the tolerance specification of a design feature determines the presence of the corresponding process planning features.

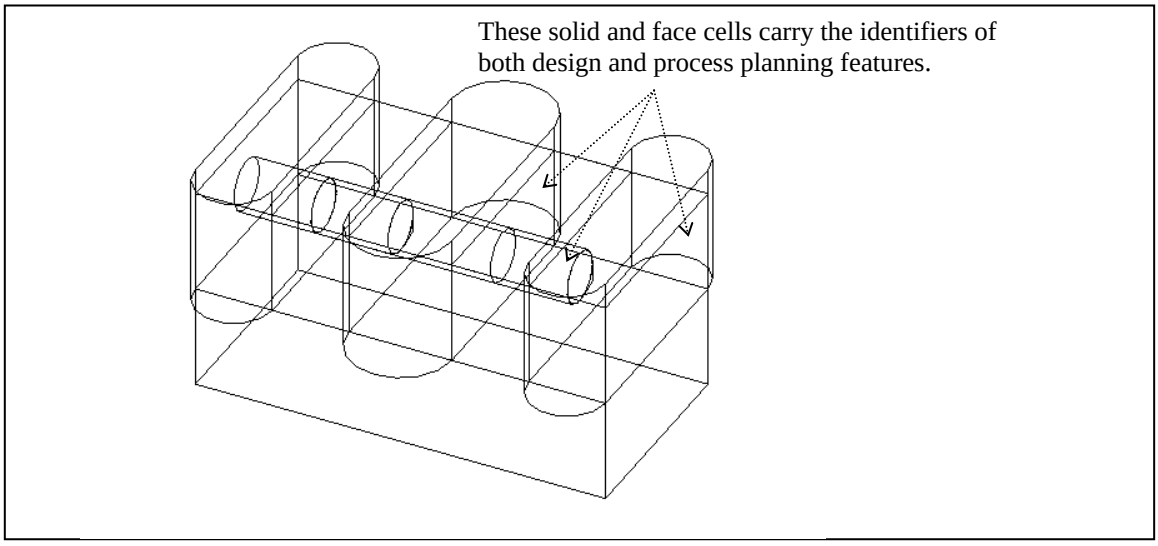


Figure 7.44 The unified cellular model in the central database

Design Changes

For example, the designer changes the diameter of the right side hole (Figure 7.45). In the design application, the associations between the old hole and its parent boss feature is retrieved from JTMS to generate the new hole feature automatically.

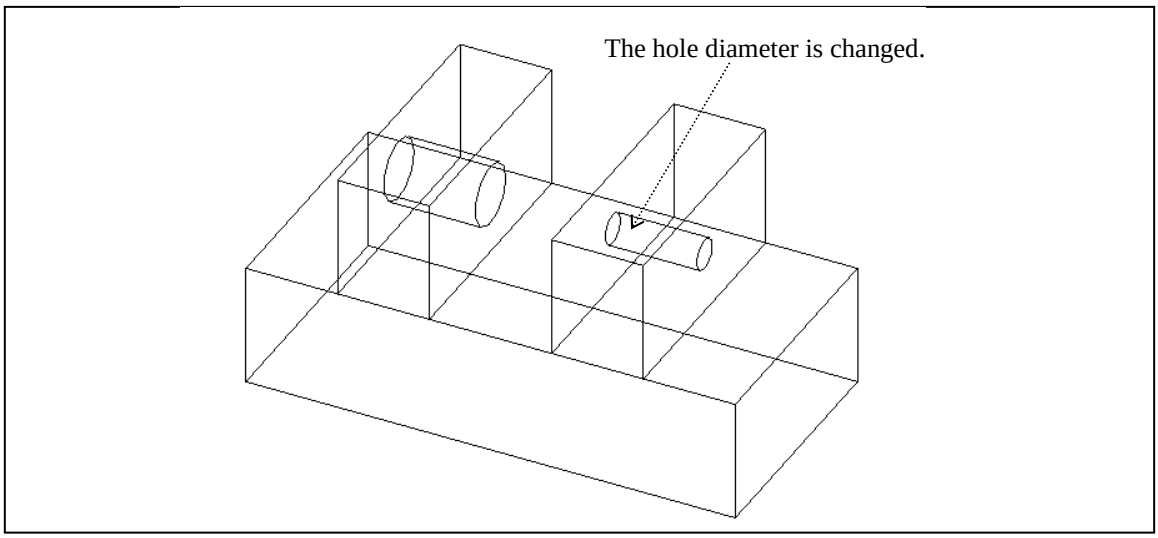


Figure 7.45 The diameter of the right side hole is changed in the design application

After modifications, because the right side cylindrical face cell in the central database carries both the design hole feature and the twist drilling feature, the drilling feature is found and the design change is propagated to the process planning application. A warning message is shown to the process planner (Figure 7.46).

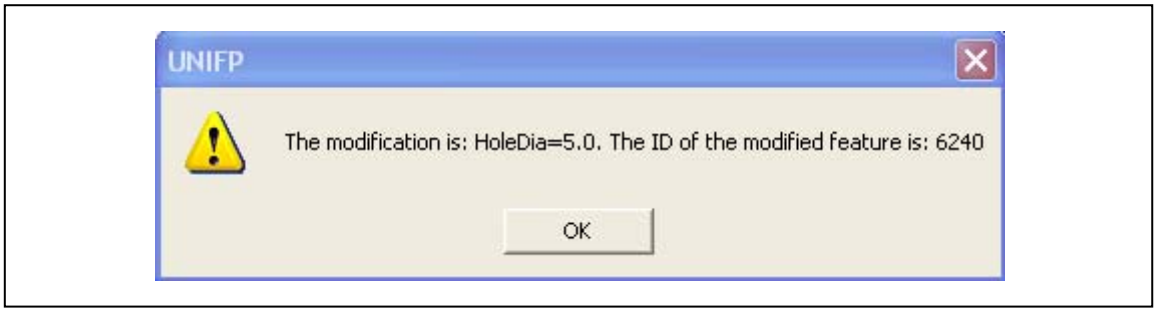


Figure 7.46 The process planner gets the design change through querying the central database

After searching JTMS, the old drilling feature is found and deleted. A new hole design feature is created first (Figure 7.47).

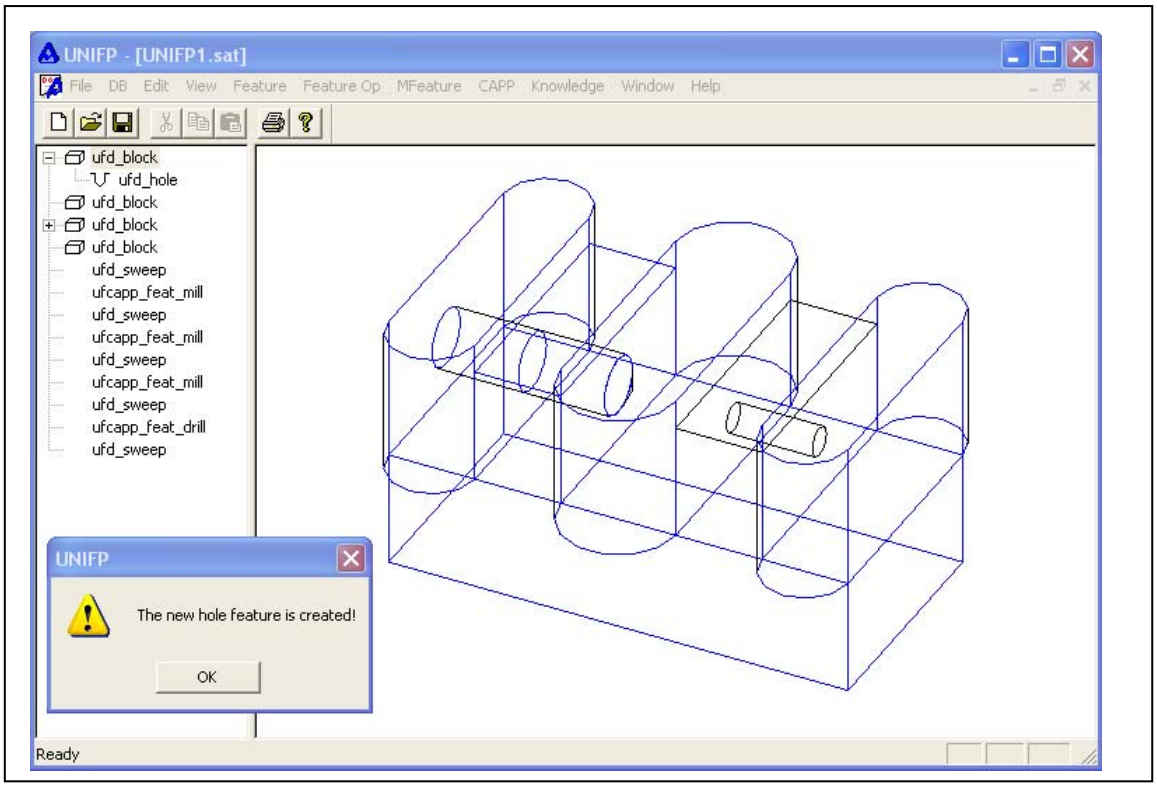


Figure 7.47 The old drilling feature is deleted and a new hole feature is created

Then, a new drilling feature is generated according to the new hole feature (Figure 7.48).

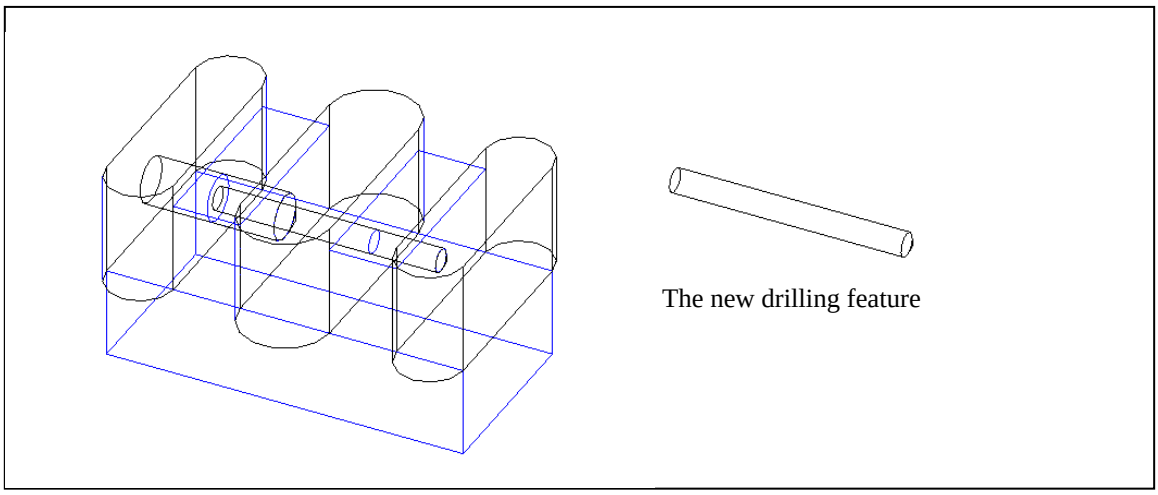


Figure 7.48 A new drilling feature is created

The associated rules and constraints to this drilling feature is found using JTMS and checked to validate the modification. It is found that the twist drilling ratio rule (by

checking the algebraic constraint) is violated because the length of the new drilling feature is too long (Figure 7.49).

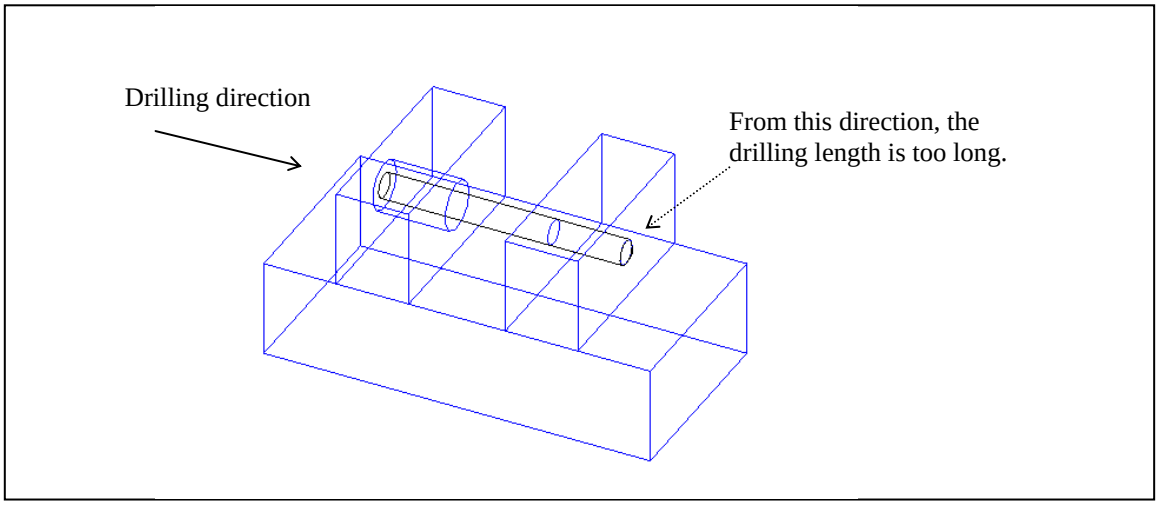


Figure 7.49 The associated algebraic constraint is violated

Through checking the associated operation element object, an alternative solution (drilling from the opposite direction) is found (Figure 7.50).

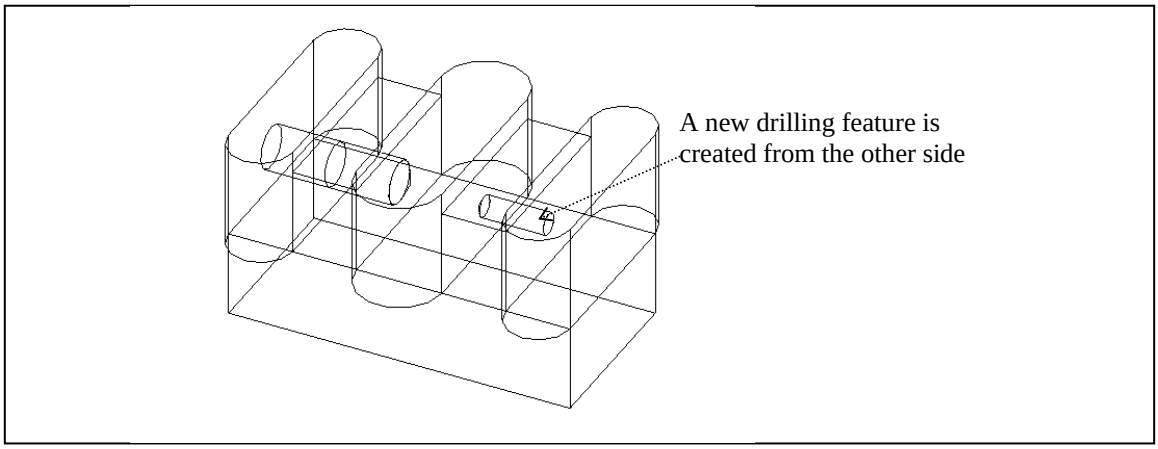


Figure 7.50 An alternative solution is found

The designer further adds a concentricity tolerance between two holes. In the process planning application, the design change is retrieved from the database. This design change is inserted into the process planning knowledge base as a new fact. A process planning rule, “if the concentricity specification between two holes is higher than a specified value, these two drilling features should be put into the same setup”, is then fired and requires putting the two corresponding drilling features into the same setup.

After consulting to the associated operation element object, no solution is found to fulfill the concentricity and the drilling ratio constraints simultaneously (Figure 7.51). The contradiction is fed back to the design application for negotiation.

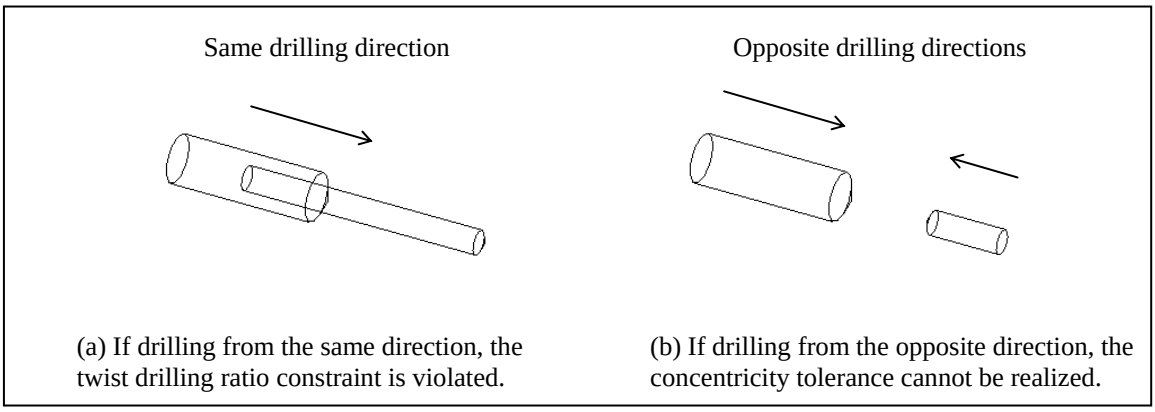


Figure 7.51 It is hard to fulfill the two design requirements simultaneously

The design application queries the database to get the failure message about the design modification. The unified feature class definition is used to retrieve the generic data in the process planning features. For example, the rules and constraints associated with the process planning features can be retrieved and displayed to the designers. The cells referred by the process planning features can be found to display the feature geometry. Therefore, the designer can get better understanding of why the design specifications are conflict from the viewpoint of process planning.

This case study shows that engineering knowledge can be embedded into computer-aided systems to justify the presence or the property values of application features.

Chapter 8
Conclusions and Future Work

8.1 Conclusions

In the context of modern concurrent and collaborative engineering, a unified product model, which covers all the product development stages, is demanded by the industry. This research aims at solving the fundamental issues of product information unification and association. A unified feature based approach is proposed. Feature-based knowledge embedment and application integration are focused.

Product development stages have much in common. These commonalities are generalized as the unified feature model and the unified cellular model. The unification facilitates data sharing. The comprehensive data sharing is necessary for maintaining the validity and consistency of product models.

The product development is an evolving decision-making process. Each decision on function structures, product specifications, or process plans is the result of a reasoning process. This reasoning process and the engineering knowledge used justify the validity of a product model. A knowledge-based system has been incorporated to manage the decision making processes throughout conceptual design, detailed design, and process planning stages. All decisions are recorded along with their justifications, which improve the maintainability of a product model greatly. These justifications and the relevant entities are represented as associations. These associations are further used to maintain the validity and consistency of product models during modifications. A justification-based truth maintenance system and a relational database are exploited to establish and manage different types of associations.

Based on the information unification and association, the proposed unified feature-based product modeling scheme establishes the basic mechanisms for the integration of multiple applications. Major technical aspects of the proposed scheme include:

- The identification of common properties and methods of different application features and a definition of these generic characteristics as unified features.

- A knowledge-based system is embedded into a feature-based geometric modeling system to handle non-geometric data and relations.
- A unified, multi-dimensional cellular model accommodates different geometric modeling requirements uniformly.
- The conceptual design stage is integrated into the whole product development management system in order to represent design intent.
- Process planning features are defined to represent detailed procedures of process planning and hence process planning strategies.
- Relations inherent to and across applications are identified for the purpose of establishing a dependency network. This dependency network, together with a change propagation algorithm, is used to check and maintain the model validity and consistency when modifications occur.
- A database structure for multiple applications to share data and to store the inter-stage associations is developed.

The feasibility of the proposed unified feature-based product modeling scheme is demonstrated by the means of a prototype system and case studies involving conceptual and detailed designs as well as process planning.

8.2 Future Work

8.2.1 Covering More Applications in the Proposed Modeling Scheme

Assembly design and CAE analysis are two important development stages for mechanical products. To accommodate these stages in the proposed scheme, the characteristics of their modeling processes need to be studied.

The possibility of applying the proposed scheme to other domains, such as electronic products, should be explored because the development of electronic products faces similar issues. For example, data exchange and associations between the design, process planning, manufacturing, and assembly stages of printed circuit board products are similarly studied (Giachetti & Alvi [202], Law et al. [203], Trappey et al. [204], ISO10303-210 [205]). Electronic products also have some different characteristics than those of mechanical products. For example, they are usually combinations of mechanical, electrical, electronic, and software systems. These characteristics bring new issues to product modeling systems, such as how to manage the different life cycles and

data models of mechanical, electronic and software systems in a consistent manner (Aberdeen [206], [46], Vossler & Dutt [207]). However, the basic idea of using the information unification and association to maintain the validity and consistency of product models remains to be suitable to electronic product development.

8.2.2 Developing a Fuzzy Rule-Based Expert System

The presented work explores the integration of rule-based expert systems with feature-based geometric modeling systems. To represent engineering intent (which is usually inexact or incomplete), it is useful to extend the current Boolean rule-based system to a fuzzy rule-based system (using fuzzy logic) to handle imprecise information. This could be achieved by using other knowledge-based technologies or theories ([181], Jackson [208]).

8.2.3 Extending the Cellular Model to Include Edge Cells and Mid-Planes

Edge cells, which are useful in representing one-dimensional features and hence useful in the embodiment design stage, need to be included. In addition, mid-planes, which are commonly used in CAE analysis, should be included in the future.

8.2.4 Improving Algorithms for Conceptual Design and Process Planning

In the current implementation, the algorithms for conceptual design and process planning are simplified as this research focuses on the theoretical exploration of the associations between knowledge, feature, and geometry. However, they have to be enhanced for industry. Issues in process planning, such as blank selection, tolerance analysis, tool path generation, 5-axis CNC machining, more flexible fixtures, need to be addressed.

This research proposes to use the conceptual design features together with functions, behaviors, and functional design rules to represent the obscure and crucial characteristics of engineering intent. Similarly, to use the conceptual design model in real applications, implementation work is needed, such as establishing more comprehensive function and behavior libraries as well as defining conceptual design features for different products.

8.2.5 Comprehensive Database Development

In the prototype system, a database is mainly used as a medium for applications to publish their data as well as for storing the inter-stage associations. The unified cellular model is stored in the database and an independent application is developed to maintain the unified cellular model. However, each application still largely relies on files. Developing a comprehensive database-driven, unified feature modeling system, is necessary for concurrent access, web-based applications, and data management (Kim & Han [209]).

The database can also support integrating the proposed product modeling scheme into the whole enterprise information system. For example, besides product data, manufacturing resources and standard components are also usually stored in the engineering database as part of the ERP. The product design, process planning, and assembly planning modules should be able to check the data, adjust their product models and production plans accordingly to achieve an overall feasible solution.

References

[1] B. Prasad. *Concurrent engineering fundamentals: integrated product and process organization*, Prentice Hall, 1996.

[2] M. A. Rosenman and J. S. Gero. Modelling multiple views of design objects in a collaborative CAD environment, *Computer-Aided Design*, Vol. 28, No. 3, pp. 193-205, 1996.

[3] D. Xue and H. Yang. A concurrent engineering-oriented design database representation model, *Computer-Aided Design*, Vol. 36, No. 10, pp. 947-965, 2004.

[4] D. Xue, S. Yadav, and D. H. Norrie. Knowledge base and database representation for intelligent concurrent design, *Computer-Aided Design*, Vol. 31, No. 2, pp. 131-145, 1999.

[5] L. Roucoules, O. Salomons, and H. Paris. Process planning as an integration of knowledge in the detailed design phase, *International Journal of Computer Integrated Manufacturing*, Vol. 16, No. 1, pp. 25-37, 2003.

[6] S. C. Feng and E. Y. Song. Information modeling of conceptual design integrated with process planning, *Proceedings of Symposia on Design For Manufacturability*, 2000.

[7] S. C. Feng and E. Y. Song. Information modeling of conceptual process planning integrated with conceptual design, *The 2000 ASME Design Engineering Technical Conferences*, 2000.

[8] J. Y. H. Fuh and W. D. Li. Advances in collaborative CAD: the-state-of-the art, *Computer-Aided Design*, Vol. 37, No. 5, pp. 571-581, 2005.

[9] H. Yang and D. Xue. Recent research on developing Web-based manufacturing systems: a review, *International Journal of Production Research*, Vol. 41, No. 15, pp. 3601-3629, 2003.

[10] L.-H. Wang, W.-M. Shen, H. Xie, J. Neelamkavil, and A. Pardasani. Collaborative conceptual design – state of the art and future trends, *Computer-Aided Design*, Vol. 34, No. 13, pp. 981-996, 2002.

[11] M. Rosenman and F.-J. Wang. CADOM: a component agent-based design-oriented model for collaborative design, *Research in Engineering Design*, Vol. 11, No. 4, pp. 193-205, 1999.

[12] F. T. S. Chan, J. Zhang, H. C. W. Lau, and A. Ning. Information integration platform for CIMS, *Proceedings of the 2000 IEEE International Conference on Management of Innovation and Technology*, 2000.

[13] F. Pahng, N. Senin, and D. Wallace. Distribution modeling and evaluation of product design problems, *Computer-Aided Design*, Vol. 30, No. 6, pp. 411-423, 1998.

[14] H. K. Kung, T. C.-T. Du, and C. H. Weng. Applying object-oriented database technologies in concurrent design processes, *International Journal of Computer Integrated Manufacturing*, Vol. 12, No. 3, pp. 251-264, 1999.

[15] W. D. Li, S. K. Ong, J. Y. H. Fuh, Y. S. Wong, Y. Q. Lu, and A. Y. C. Nee. Feature-based design in a distributed and collaborative environment, *Computer-Aided Design*, Vol. 36, No. 9, pp. 775-797, 2004.

[16] R. Bidarra, E. van den Berg, and W. F. Bronsvoort. Collaborative modeling with features, *Proceedings of the 2001 ASME Design Engineering Technical Conferences*, 2001.

[17] J. Y. Lee, S. B. Han, H. Kim, and S. B. Park. Network-centric feature-based modeling, *Proceedings of the Seventh Pacific Conference on Computer Graphics and Applications*, 1999.

[18] N. Shyamsundar and R. Gadh. Collaborative virtual prototyping of product assemblies over the Internet, *Computer-Aided Design*, Vol. 34, No. 10, pp. 755-768, 2002.

[19] T. Varady, R. R. Martin, and J. Cox. Reverse engineering of geometric models – an introduction, *Computer-Aided Design*, Vol. 29, No. 4, pp. 255-268, 1997.

[20] P. Benko, R. R. Martin, and T. Varady. Algorithms for reverse engineering boundary representation models, *Computer-Aided Design*, Vol. 33, No. 11, pp. 839-851, 2001.

[21] B. Starly, A. Lau, W. Sun, W. Lau, and T. Bradbury. Direct slicing of STEP based NURBS models for layered manufacturing, *Computer-Aided Design*, Vol. 37, No. 4, pp. 387-397, 2005.

[22] T. R. Kramer, H. Huang, E. Messina, F. M. Proctor, and H. Scott. A feature-based inspection and machining system, *Computer-Aided Design*, Vol. 33, No. 9, pp. 653-669, 2001.

[23] M. Rezayat. Midsurface abstraction from 3D solid models: general theory and applications, *Computer-Aided Design*, Vol. 28, No. 11, pp. 905-915, 1996.

[24] W.-Y. Ma, Y.-M. Zhong, S.-K. Tso, and T.-X. Zhou. A hierarchically structured and constraint-based data model for intuitive and precise solid modeling in a virtual reality environment, *Computer-Aided Design*, Vol. 36, No. 10, pp. 903-928, 2004.

[25] J. Y. H. Fuh, C.-H. Chang, and M. A. Melkanoff. The development of an integrated and intelligent CAD/CAPP/CAFP environment using logic-based reasoning, *Computer-Aided Design*, Vol. 28, No. 3, pp. 217-232, 1996.

[26] A. Noort, G. F. M. Hoek, and W. F. Bronsvoort. Integrating part and assembly modeling, *Computer-Aided Design*, Vol. 34, No. 12, pp. 899-912, 2002.

[27] Industrial automation systems and integration: product data representation and exchange: Integrated generic resource: Part 42 – Geometric and topological representation, *International Organization for Standardization (ISO)*, Geneva, Switzerland, 2000.

[28] R. E. Giachetti. A framework to review the information integration of the enterprise, *International Journal of Production Research*, Vol. 42, No. 6, pp. 1147-1166, 2004.

[29] E. J. Umble, R. R. Haft, and M. M. Umble. Enterprise resource planning: implementation procedures and critical success factors, *European Journal of Operational Research*, Vol. 146, No. 2, pp. 241–257, 2003.

[30] F. R. Jacobs and E. Bendoly. Enterprise resource planning: developments and directions for operations management research, *European Journal of Operational Research*, Vol. 146, No. 2, pp. 233–240, 2003.

[31] H. A. Akkermans, P. Bogerd, E. Yücesan, and L. N. van Wassenhove. The impact of ERP on supply chain management: exploratory findings from a European Delphi study, *European Journal of Operational Research*, Vol. 146, No. 2, pp. 284–301, 2003.

[32] K. Park and A. Kusiak. Enterprise resource planning (ERP) operations support system for maintaining process integration, *International Journal of Production Research*, Vol. 43, No. 19, pp. 3959–3982, 2005.

[33] C.-B. Wang, T.-Y. Chen, Y.-M. Chen, and H.-C. Chu. Design of a meta model for integrating enterprise systems, *Computers in Industry*, Vol. 56, No. 3, pp. 305-322, 2005.

[34] C. Ou-Yang and M. J. Chang. Developing an agent-based PDM/ERP collaboration system, *International Journal of Advanced Manufacturing Technology*, Vol. 30, No. 3-4, pp. 369–384, 2006.

[35] S. B. Tor, G. A. Britton, W. Y. Zhang, and Y.-M. Deng. Guiding functional design of mechanical products through rule-based causal behavioural reasoning, *International Journal of Production Research*, Vol. 40, No. 3, pp. 667-682, 2002.

[36] J. Y. Lee and K. Kim. Geometric reasoning for knowledge-based parametric design using graph representation, *Computer-Aided Design*, Vol. 28, No. 10, pp. 831-841, 1996.

[37] S. Myung and S. Han. Knowledge-based parametric design of mechanical products based on configuration design method, *Expert Systems with Applications*, Vol. 21, No. 2, pp. 99-107, 2001.

[38] F. Zhang and D. Xue. Distributed database and knowledge base modeling for concurrent design, *Computer-Aided Design*, Vol. 34, No. 1, pp. 27-40, 2002.

[39] X.-F. Zha, H.-J. Du, and J.-H. Qiu. Knowledge-based approach and system for assembly oriented design, Part I: the approach, *Engineering Applications of Artificial Intelligence*, Vol. 14, No. 1, pp. 61-75, 2001.

[40] X.-F. Zha, H.-J. Du, and J.-H. Qiu. Knowledge-based approach and system for assembly-oriented design, Part II: the system implementation, *Engineering Applications of Artificial Intelligence*, Vol. 14, No. 2, pp. 239-254, 2001.

[41] M. R. Henderson. *Extraction of Feature Information from Three-Dimensional CAD Data*, Ph.D. Dissertation, Purdue University, West Lafayette, IN, USA, 1984.

[42] R.-S. Lee, Y.-M. Chen, and C.-Z. Lee. Development of a concurrent mold design system: a knowledge-based approach, *Computer Integrated Manufacturing Systems*, Vol. 10, No. 4, pp. 287-307, 1997.

[43] C. K. Mok, K. S. Chin, and J. K. L. Ho. An interactive knowledge-based CAD system for mould design in injection moulding processes, *International Journal of Advanced Manufacturing Technology*, Vol. 17, No. 1, pp. 27-38, 2001.

[44] D. N. Sormaz and B. Khoshnevis. Process planning knowledge representation using an object-oriented data model, *International Journal of Computer Integrated Manufacturing*, Vol. 10, No. 1-4, pp. 92-104, 1997.

[45] S. C. Park. Knowledge capturing methodology in process planning, *Computer-Aided Design*, Vol. 35, No. 12, pp. 1109-1117, 2003.

[46] http://www.ugs.com

[47] http://www.cocreate.com

[48] Y.-S. Ma. A case study on non-parametric design method in ODM collaborative product development, *International Journal of Product Development*, Vol. 2, No. 4, pp. 411-434, 2005.

[49] M. J. Pratt, B. D. Anderson, and T. Ranger. Towards the standardized exchange of parameterized feature-based CAD models, *Computer-Aided Design*, Vol. 37, No. 12, pp. 1251-1265, 2005.

[50] J. J. Shah and M. Mantyla. *Parametric and feature-based CAD/CAM: concepts, techniques, and applications*, John Wiley and Sons, Inc., 1995.

[51] P. R. Wilson and M. J. Pratt. A taxonomy of features for solid modeling, In: *Geometric modeling for CAD applications*, M. J. Wozny, H. W. McLaughlin and J. L. Encarnacao (Eds.), North-Holland, pp. 125-136, 1988.

[52] J. J. Cunningham and J. R. Dixon. Designing with features: the origin of features, *Proceedings of ASME Computers in Engineering Conference*, San Francisco, pp. 237-243, 1988.

[53] J. J. Shah. Conceptual Development of Form Features and Feature Modelers, *Research in Engineering Design*, Vol. 2, No. 2, pp. 93-108, 1991.

[54] J. H. Vandenbrande and A. A. G. Requicha. Spatial reasoning for the automatic recognition of machinable features in solid models, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 12, pp. 1269-1285, 1993.

[55] T. D. Martino, B. Falcidieno, and S. Habinger. Design and engineering process integration through a multiple view intermediate modeler in a distributed object-oriented system environment, *Computer-Aided Design*, Vol. 30, No. 6, pp. 437-452, 1998.

[56] R. Bidarra and W. F. Bronsvoort. Semantic feature modeling, *Computer-Aided Design*, Vol. 32, No. 3, pp. 201-225, 2000.

[57] C. M Hoffman. *Geometric and solid modeling: an introduction*, Morgan Kaufmann, 1989.

[58] A. A. G. Requicha. Representations for rigid solids: theory, methods, and systems, *ACM Computing Surveys (CSUR)*, Vol. 12, No. 4, pp. 437-464, 1980.

[59] S. Venkataraman, J. J. Shah, and J. D. Summers. An investigation of integrating design by features and feature recognition, *IFIP Conference, FEATS 2001*, Valenciences, Frances, 2001.

[60] U. Roy and C. R. Liu. Feature-based representational scheme of a solid modeler for providing dimensioning and tolerancing information, *Robotics & Computer-Integrated Manufacturing*, Vol. 4, No. 3/4, pp. 335-345, 1988.

[61] D. C. Gossard, R. R. Zuffante, and H. Sakurai. Representing dimensions, tolerances, and features in MCAE systems, *IEEE Computer Graphics and Applications*, Vol. 8, No. 2, pp. 51-59, 1988.

[62] H. E. Otto. From concepts to consistent object specifications: translation of a domain-oriented feature framework into practice, *Journal of computer science & technology*, Vol. 16, No. 3, pp. 208-230, 2001.

[63] G. Brunetti, T. De Martino, B. Falcidieno, and S. HaBinger. A relational model for interactive manipulation of form features based on algebraic geometry, *Proceedings of the third ACM symposium on Solid modeling and applications*, Salt Lake City, Utah, USA, 1995.

[64] J. J. Shah and M. T. Rogers. Assembly modeling as an extension of feature-based design, *Research in Engineering Design*, Vol. 5, No. 3-4, pp. 218-237, 1993.

[65] M. S. Hounsell and K. Case. Feature-based interaction: an identification and classification methodology, *Proceedings of the Institution of Mechanical Engineers, Part B – Journal of Engineering Manufacture*, Vol. 213, No. 4, pp. 369-380, 1999.

[66] R. Bidarra, M. Dohmen, and W. F. Bronsvoort. Automatic detection of interactions in feature models, *Proceedings of the 1997 ASME Design Engineering Technical Conferences*, 1997.

[67] R. R. Karinthi and D. Nau. An algebraic approach to feature interactions, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14, No. 4, pp. 469-484, 1992.

[68] W. Faheem, C. C. Hayes, J. F. Castano, and D. M. Gaines. What is a manufacturing interaction? *Proceedings of the 1998 ASME Design Engineering Technical Conferences*, 1998.

[69] W. C. Regli and M. J. Pratt. What are feature interactions? *Proceedings of the 1996 ASME Design Engineering Technical Conference and Computers in Engineering Conference*, 1996.

[70] A. Mukherjee and C. R. Liu. Conceptual design, manufacturability evaluation and preliminary process planning using function-form relationships in stamped metal parts, *Robotics & Computer-Integrated Manufacturing*, Vol. 13, No. 3, pp. 253-270, 1997.

[71] C.-X. Feng, C.-C. Huang, A. Kusiak, and P.-G. Li. Representation of functions and features in detail design, *Computer-Aided Design*, Vol. 28, No. 12, pp. 961-971, 1996.

[72] D. E. Whitney, R. Mantripragada, J. D. Adams, and S. J. Rhee. Designing assemblies, *Research in Engineering Design*, Vol. 11, No. 4, pp. 229-253, 1999.

[73] G. Brunetti and B. Golob. A feature-based approach towards an integrated product model including conceptual design information, *Computer-Aided Design*, Vol. 32, No. 14, pp. 877-887, 2000.

[74] G. Brunetti and S. Grimm. Feature ontologies for the explicit representation of shape semantics, *International Journal of Computer Applications in Technology*, Vol. 23, No. 2/3/4, pp. 192-202, 2005.

[75] D. N. Sormaz and B. Khoshnevis. Generation of alternative process plans in integrated manufacturing systems, *Journal of Intelligent Manufacturing*, Vol. 14, No. 6, pp. 509-526, 2003.

[76] B. Khoshnevis, D. N. Sormaz, and J. Y. Park. An integrated process planning system using feature reasoning and space search-based optimization, *IIE Transactions*, Vol. 31, No. 7, pp. 597-616, 1999.

[77] J. R. Rossignac. Issues on feature-based editing and interrogation of solid models, *Computers & Graphics*, Vol. 14, No. 2, pp. 149-172, 1990.

[78] R. Geelink, O. W. Salomons, F. van Slooten, F. J. A. M. van Houten, and H. J. J. Kals. Unified feature definition for feature based design and feature based manufacturing, *Proceedings of the ASME International Computers in Engineering Conference*, 1995.

[79] T. Laakko and M. Mantyla. Feature modeling by incremental feature recognition, *Computer-Aided Design*, Vol. 25, No. 8, pp. 479-492, 1993.

[80] F. Mandorli, U. Cugini, H. E. Otto, and F. Kimura. Modeling with self validation features, *Proceedings of ACM/IEEE Symposium on Solid Modeling and Applications'97*, pp. 88-96, 1997.

[81] A. S. Vieira. Consistency management in feature-based parametric design, *Proceedings of the 1995 ASME Design Engineering Technical Conferences*, Vol. 2, pp. 977-987, 1995.

[82] T. D. Martino, B. Falcidieno, F. Giannini, S. Hassinger, and J. Ovtcharova. Feature-based modeler by integrating design and recognition approaches, *Computer-Aided Design*, Vol. 26, No. 8, pp. 646-653, 1994.

[83] V. Sharma and C. C. Hayes. Operation ordering principles and intra-setup planner: combining human control with automation in process planning, *Proceedings of the 2001 ASME Design Engineering Technical Conferences*, 2001.

[84] R. Anderl and R. Mendgen. Modelling with constraints: theoretical foundation and application, *Computer-Aided Design*, Vol. 28, No. 3, pp. 301-313, 1996.

[85] A. Oral and M. C. Cakir. Automated cutting tool selection and cutting tool sequence optimization for rotational parts, *Robotics & Computer-Integrated Manufacturing*, Vol. 20, No. 2, pp. 127-141, 2004.

[86] D. C. Brown. Functional, behavioral and structural features, *Proceedings of KIC5, 5th IFIP WG5.2 Workshop on Knowledge Intensive CAD*, 2002.

[87] B. D. Mcginnis and D. G. Ullman. The evolution of commitments in the design of a component, *Journal of Mechanical Design*, Vol. 114, No. 1, pp. 1-7, 1992.

[88] S. L. Wood and D. G. Ullman. The functions of plastic injection moulding features, *Design Studies*, Vol. 17, No. 2, pp. 201-213, 1996.

[89] M. Schulte, C. Weber, and R. Stark. Functional features for design in mechanical engineering, *Computers in Industry*, Vol. 23, No. 1-2, pp. 15-24, 1993.

[90] S. H. Lee. A CAD-CAE integration approach using feature-based multi-resolution and multi-abstraction modeler techniques, *Computer-Aided Design*, Vol. 37, No. 9, pp. 941-955, 2005.

[91] Y.-S. Ma and T. Tong. Associative feature modeling for concurrent engineering integration, *Computers in Industry*, Vol. 51, No. 1, pp. 51-71, 2003.

[92] P. D. Stefano, F. Bianconi, and L. D. Angelo. An approach for feature semantics recognition in geometric models, *Computer-Aided Design*, Vol. 36, No. 10, pp. 993-1009, 2004.

[93] G. Boothroyd, P. Dewhurst, and W. Knight. *Product design for manufacture and assembly*, 2nd Edition, Marcel Dekker, Inc., 2002.

[94] T. L. D. Fazio, S. J. Rhee, and D. E. Whitney. Design-specific approach to design for assembly (DFA) for complex mechanical assemblies, *IEEE Transactions on Robotics and Automation*, Vol. 15, No. 5, pp. 869-881, 1999.

[95] S. K. Ong and L. C. Chew. Evaluating the manufacturability of machined parts and their setup plans, *International Journal of Production Research*, Vol. 38, No. 11, pp. 2397-2415, 2000.

[96] C. L. Li. Automatic layout design of plastic injection mould cooling system, *Computer-Aided Design*, Vol. 37, No. 7, pp. 645-662, 2005.

[97] L. Chen, J. Pu, and X.-K. Wang. A general model for machinable features and its application to machinability evaluation of mechanical parts, *Computer-Aided Design*, Vol. 34, No. 3, pp. 239-249, 2002.

[98] H. L. Lockett and M. D. Guenov. Graph-based feature recognition for injection moulding based on a mid-surface approach, *Computer-Aided Design*, Vol. 37, No. 2, pp. 251-262, 2005.

[99] J. H. Han and A. A. G. Requicha. Feature recognition from CAD models, *IEEE Computer Graphics and Applications*, Vol. 18, No. 2, pp. 80-94, 1998.

[100] Y.-M. Deng, G. A. Britton, Y. C. Lam, S. B. Tor, and Y.-S. Ma. Feature-based CAD-CAE integration model for injection-moulded product design, *International Journal of Production Research*, Vol. 40, No. 15, pp. 3737-3750, 2002.

[101] C. K. Chan and S. T. Tan. Generating assembly features onto split solid models, *Computer-Aided Design*, Vol. 35, No. 14, pp. 1315-1336, 2003.

[102] K. Case and W. A. W. Harun. Feature-based representation for manufacturing planning, *International Journal of Production Research*, Vol. 38, No. 17, pp. 4285-4300, 2000.

[103] R. Anantha, G. A. Kramer, and R. H. Crawford. Assembly modeling by geometric constraint satisfaction, *Computer-Aided Design*, Vol. 28, No. 9, pp. 707-722, 1996.

[104] Y.-S. Ma, G. A. Britton, S. B. Tor, and L.-Y. Jin. Associative assembly design features: concept, implementation and application, accepted by *International Journal of Advanced Manufacturing Technology*.

[105] A. Csabai, I. Stroud, and P. C. Xirouchakis. Container spaces and functional features for top-down 3D layout design, *Computer-Aided Design*, Vol. 34, No. 13, pp. 1011-1035, 2002.

[106] N. Shyamsundar and R. Gadh. Internet-based collaborative product design with assembly features and virtual design spaces, *Computer-Aided Design*, Vol. 33, No. 9, pp. 637-651, 2001.

[107] D. Deneux. Introduction to assembly features: an illustrated synthesis methodology, *Journal of Intelligent Manufacturing*, Vol. 10, No. 1, pp. 29-39, 1999.

[108] Industrial automation systems and integration: product data representation and exchange: application protocol: Part 224 – mechanical product definition for process planning using machining features, *International Organization for Standardization (ISO)*, Geneva, Switzerland, 1999.

[109] S. Joshi and T. C. Chang. Graph-based heuristics for recognition of machined features from a 3D solid model, *Computer-Aided Design*, Vol. 20, No. 2, pp. 58-66, 1988.

[110] Y.-J. Tseng and S. B. Joshi. Recognizing multiple interpretations of interacting machining features, *Computer-Aided Design*, Vol. 26, No. 9, pp. 667-688, 1994.

[111] S. K. Gupta and D. S. Nau. Systematic approach to analyzing the manufacturability of machined parts, *Computer-Aided Design*, Vol. 27, No. 5, pp. 323-342, 1995.

[112] C.-C. P. Chu and R. Gadh. Feature-based approach for set-up minimization of process design from product design, *Computer-Aided Design*, Vol. 28, No. 5, pp. 321-332, 1996.

[113] M. Marefat and J. Britanik. Case-based process planning using an object-oriented model representation, *Robotics & Computer-Integrated Manufacturing*, Vol. 13, No. 3, pp. 229-251, 1997.

[114] A. S. Kumar, A. Y. C. Nee, and S. Prombanpong. Expert fixture-design system for an automated manufacturing environment, *Computer-Aided Design*, Vol. 24, No. 6, pp. 316-326, 1992.

[115] P. D. Stefano. Automatic extraction of form features for casting, *Computer-Aided Design*, Vol. 29, No. 11, pp. 761-770, 1997.

[116] W. van Holland and W. F. Bronsvoort. Assembly features in modeling and planning, *Robotics & Computer Integrated Manufacturing*, Vol. 16, No. 4, pp. 277-294, 2000.

[117] K.-Y. Kim, D. G. Manley, and H. J. Yang. Ontology-based assembly design and information sharing for collaborative product development, *Computer-Aided Design*, Vol. 38, No. 12, pp. 1233-1250, 2006.

[118] K.-Y. Kim, Y. Wang, O. S. Muogboh, and B. O. Nnaji. Design formalism for collaborative assembly design, *Computer-Aided Design*, Vol. 36, No. 9, pp. 849-871, 2004.

[119] M. Dohmen. *Constraint-based feature validation*, Ph.D. Dissertation, Delft University of Technology, Netherlands, 1997.

[120] J. Y. Park and B. Khoshnevis. A real-time computer-aided process planning system as a support tool for economic product design, *Journal of Manufacturing Systems*, Vol. 12, No. 2, pp. 181-193, 1993.

[121] J. Gao, D.-T. Zheng, and N. Gindy. Mathematical representation of feature conversion for CAD/CAM system integration, *Robotics & Computer-Integrated Manufacturing*, Vol. 20, No. 5, pp. 457-467, 2004.

[122] W. F. Bronsvoort and F. W. Jansen. Feature modelling and conversion – Key concepts to concurrent engineering, *Computers in Industry*, Vol. 21, No. 1, pp. 61-86, 1993.

[123] W. F. Bronsvoort and A. Noort. Multiple-view feature modeling for integral product development, *Computer-Aided Design*, Vol. 36, No. 10, pp. 929-946, 2004.

[124] D. C. Anderson and T. C. Chang. Geometric reasoning in feature-based design and process planning, *Computers & Graphics*, Vol. 14, No. 2, pp. 225-235, 1990.

[125] S. Subramani and B. Gurumoorthy. Maintaining associativity between form feature models, *Computer-Aided Design*, Vol. 37, No. 13, pp. 1319-1334, 2005.

[126] S. Subramani, S. R. P. R. Nalluri, and B. Gurumoorthy. 3D clipping algorithm for feature mapping across domains, *Computer-Aided Design*, Vol. 36, No. 8, pp. 701-721, 2004.

[127] J. J. Shah. Feature transformations between application-specific feature spaces, *Journal of Computer-Aided Engineering*, Vol. 5, No. 6, pp. 247-255, 1988.

[128] P. Pal, A. M. Tigga, and A. Kumar. Feature extraction from large CAD databases using genetic algorithm, *Computer-Aided Design*, Vol. 37, No. 5, pp. 545-558, 2005.

[129] Y. S. Kim. Volumetric feature recognition using convex decomposition, In: *Advances in Feature Based Manufacturing, Manufacturing Research and Technology*, J. J. Shah, M. Mantyla and D. S. Nau (Eds.). Elsevier Science, pp. 39-63, 1994.

[130] W. C. Regli. *Geometric algorithms for recognition of features from solid models*, Ph.D. Dissertation, University of Maryland, College Park, MD, USA, 1994.

[131] R. Raman and M. M. Marefat. Integrated process planning using tool/process capabilities and heuristic search, *Journal of Intelligent Manufacturing*, Vol. 15, No. 2, pp. 141-174, 2004.

[132] W. D. Li, S. K. Ong, and A. Y. C. Nee. Recognizing manufacturing features from a design-by-feature model, *Computer-Aided Design*, Vol. 34, No. 11, pp. 849-868, 2002.

[133] D. M. Gaines and C. C. Hayes. CUSTOM-CUT: a customizable feature recognizer, *Computer-Aided Design*, Vol. 31, No. 2, pp. 85-100, 1999.

[134] R. Stage, C. Roberts, and M. Henderson. Generating resource based flexible form manufacturing features through objective driven clustering, *Computer-Aided Design*, Vol. 31, No. 2, pp. 119-130, 1999.

[135] J. H. Han. *3D geometric reasoning algorithms for feature recognition*, Ph.D. Dissertation, University of Southern California, Los Angeles, CA, USA, 1996.

[136] F. Zhou, T.-C. Kuo, S. H. Huang, and H.-C. Zhang. Form feature and tolerance transfer from a 3D model to a set-up planning system, *International Journal of Advanced Manufacturing Technology*, Vol. 19, No. 2, pp. 88-96, 2002.

[137] K. Jha and B. Gurumoorthy. Automatic propagation of feature modification across domains, *Computer-Aided Design*, Vol. 32, No. 12, pp. 691-706, 2000.

[138] R. Bidarra, K. J. de Kraker, and W. F. Bronsvoort. Representation and management of feature information in a cellular model, *Computer-Aided Design*, Vol. 30, No. 4, pp. 301-313, 1998.

[139] M. Dohmen, K. J. de Kraker, and W. F. Bronsvoort. Feature validation in a multiple-view modeling system, *Proceedings of the 1996 ASME Design Engineering Technical Conference and Computers in Engineering Conference*, 1996.

[140] K. J. de Kraker, M. Dohmen, and W. F. Bronsvoort. Maintaining multiple views in feature modeling, *The 4th Symposium on Solid Modeling and Applications*, ACM Press, pp. 123-130, 1997.

[141] Y. S. Suh and M. J. Wozny. Interactive Feature Extraction for a Form Feature Conversion System, *The 4th Symposium on Solid Modeling and Applications*, ACM Press, pp. 111-122, 1997.

[142] M. J. Pratt and V. Srinivasan. Towards a neutral specification of geometric features, *International Journal of Computer Applications in Technology*, Vol. 23, No. 2/3/4, pp. 203-218, 2005.

[143] Y.-S. Ma and T. Tong. An object-oriented design tool for associative cooling channels in plastic-injection moulds, *International Journal of Advanced Manufacturing Technology*, Vol. 23, No. 1-2, pp. 79-86, 2004.

[144] Y.-S. Ma, S. B. Tor, and G. A. Britton. The development of a standard component library for plastic injection mould design using an object-oriented approach, *International Journal of Advanced Manufacturing Technology*, Vol. 22, No. 9-10, pp. 611-618, 2003.

[145] Y.-S. Ma, G. A. Britton, S. B. Tor, L.-Y. Jin, G. Chen, and S.-H. Tang. Design of a feature-object-based mechanical assembly library, *Computer-Aided Design & Applications*, Vol. 1, No. 1-4, pp. 397-404, 2004.

[146] J. J. Shah, A. Ali, and M. T. Rogers. Investigation of declarative feature modeling, *Proceedings of the ASME'94 Computers in Engineering*, Vol. 1, pp. 1-11, 1994.

[147] J.-K Gui and M. Mantyla. Functional understanding of assembly modeling, *Computer-Aided Design*, Vol. 26, No. 6, pp. 435-451, 1994.

[148] M. R. Henderson. Representing functionality and design intent in product models, *Proceedings of the second ACM symposium on Solid modeling and applications*, Montreal, Canada, 1993.

[149] B. Chandrasekaran, A. K. Goel, and Y. Iwasaki. Functional representation as design rationale, *Computer*, Vol. 26, No. 1, pp. 48-56, 1993.

[150] A. Kusiak, E. Szczerbicki, and K. Park. A novel approach to decomposition of design specifications and search for solutions, *International Journal of Production Research*, Vol. 29, No. 7, pp. 1391-1406, 1991.

[151] Y. Umeda, M. Ishii, M. Yoshioka, Y. Shimomura, and T. Tomiyama. Supporting conceptual design based on the function-behavior-state modeler, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 10, No. 4, pp. 275-288, 1996.

[152] L. Qian and J. S. Gero. Function-behavior-structure paths and their role in analogy-based design, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 10, No. 4, pp. 289-312, 1996.

[153] R. V. Welch and J. R. Dixon. Representing function, behavior and structure during conceptual design, *Design Theory and Methodology – DTM'92*, pp. 11-18, 1992.

[154] X. Guan, A. H. B. Duffy, and K. J. Maccallum. Prototype system for supporting the incremental modelling of vague geometric configurations, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 11, No. 4, pp. 287-310, 1997.

[155] A. Wong and D. Sriram. SHARED: an information model for cooperative product development, *Research in Engineering Design*, Vol. 5, No. 1, pp. 21-39, 1993.

[156] G. Thimm, G. A. Britton, and S. C. Fok. A graph theoretic approach linking design dimensioning and process planning Part 1: Designing to process planning, *International Journal of Advanced Manufacturing Technology*, Vol. 24, No. 3-4, pp. 261-271, 2004.

[157] G. Thimm, G. A. Britton, and S. C. Fok. A graph theoretic approach linking design dimensioning and process planning Part 2: Design heuristics for rotational parts, *International Journal of Advanced Manufacturing Technology*, Vol. 24, No. 3-4, pp. 272-278, 2004.

[158] K. Weiler. The radial edge structure: a topological representation for non-manifold geometric boundary modeling, In: *Geometric modeling for CAD applications*, M. J. Wozny, H. W. McLaughlin and J. L. Encarnacao (Eds.), North-Holland, pp. 3-36, 1988.

[159] K. Weiler. Boundary graph operators for non-manifold geometric modeling topology representations, In: *Geometric modeling for CAD applications*, M. J. Wozny, H. W. McLaughlin and J. L. Encarnacao (Eds.), North-Holland, pp. 37-66, 1988.

[160] H. Masuda. Topological operators and Boolean operations for complex-based nonmanifold geometric models, *Computer-Aided Design*, Vol. 25, No. 2, pp. 119-129, 1993.

[161] G. A. Crocker and W. F. Reinke. An editable nonmanifold boundary representation, *IEEE Computer Graphics & Applications*, Vol. 11, No. 2, pp. 39-51, 1991.

[162] R. D. Sriram, A. Wong, and L.-X. He. GNOMES: an object-oriented nonmanifold geometric engine, *Computer-Aided Design*, Vol. 27, No. 11, pp. 853-868, 1995.

[163] R. Bidarra, J. Madeira, W. J. Neels, and W. F. Bronsvoort. Efficiency of boundary evaluation for a cellular model, *Computer-Aided Design*, Vol. 37, No. 12, pp. 1266-1284, 2005.

[164] J. Y. Lee, J.-H Lee, H. Kim, and H.-S. Kim. A cellular topology-based approach to generating progressive solid models from feature-centric models, *Computer-Aided Design*, Vol. 36, No. 3, pp. 217-229, 2004.

[165] D. Wu and R. Sarma. Dynamic segmentation and incremental editing of boundary representations in a collaborative design environment, *Proceedings of the sixth ACM symposium on Solid modeling and applications*, Ann Arbor, Michigan, United States, pp. 289-300, 2001.

[166] Y. Woo. Fast cell-based decomposition and applications to solid modeling, *Computer-Aided Design*, Vol. 35, No. 11, pp. 969-977, 2003.

[167] A. Kusiak and J. Wang. Dependency analysis in constraint negotiation, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 25, No. 9, 1301-1313, 1995.

[168] H. Park and M. R. Cutkosky. Framework for modeling dependencies in collaborative engineering processes, *Research in Engineering Design*, Vol. 11, No. 2, pp. 84-102, 1999.

[169] C. M. Eastman. Managing integrity in design information flows, *Computer-Aided Design*, Vol. 28, No. 6/7, pp. 551-565, 1996.

[170] S. R. Gorti and R. D. Sriram. From symbol to form: a framework for conceptual design, *Computer-Aided Design*, Vol. 28, No. 11, pp. 853-870, 1996.

[171] M. Ranta, M. Mantyla, Y. Umeda, and T. Tomiyama. Integration of functional and feature-based product modeling – the IMS/GNOSIS experience, *Computer-Aided Design*, Vol. 28, No. 5, pp. 371-381, 1996.

[172] U. Roy and B. Bharadwaj. Design with part behaviors: behavior model, representation and applications, *Computer-Aided Design*, Vol. 34, No. 9, pp. 613-636, 2002.

[173] U. Roy, N. Pramanik, R. Sudarsan, R. D. Sriram, and K. W. Lyons. Function-to-form mapping: model, representation and applications in design synthesis, *Computer-Aided Design*, Vol. 33, No. 10, pp. 699-719, 2001.

[174] C. M Hoffman and R. Joan-Arinyo. Distributed maintenance of multiple product views, *Computer-Aided Design*, Vol. 32, No. 7, pp. 421-431, 2000.

[175] C. M Hoffman and R. Joan-Arinyo. CAD and the product master model, *Computer-Aided Design*, Vol. 30, No. 11, pp. 905-918, 1998.

[176] D. Roller and I. Kreuz. Selecting and parameterising components using knowledge based configuration and a heuristic that learns and forgets, *Computer-Aided Design*, Vol. 35, No. 12, pp. 1085-1098, 2003.

[177] J. A. Penoyer, G. Burnett, D. J. Fawcett, and S.-Y. Liou. Knowledge based product life cycle systems: principles of integration of KBE and C3P, *Computer-Aided Design*, Vol. 32, No. 5-6, pp. 311-320, 2000.

[178] K. D. Forbus and J. de Kleer. *Building problem solvers*, MIT Press, 1993.

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

[180] G. Chen, Y.-S. Ma, G. Thimm, and S.-H. Tang. Unified feature modeling scheme for the integration of CAD and CAx, *Computer-Aided Design & Applications*, Vol. 1, No. 1-4, pp. 595-602, 2004.

[181] J. Giarratano and G. Riley. *Expert systems: principles and programming*, 4th Edition, PWS Publishing Company, 2004.

[182] E.F. Hill. *Jess in action: rule-based systems in Java*, Manning Publications Co., 2003.

[183] J.P. Bigus and J. Bigus. *Constructing intelligent agents using Java: professional developer's guide*, 2nd Edition, John Wiley & Sons, Inc., 2001.

[184] C. Lottaz, I. F. C. Smith, Y. R. Nicoud, and B. V. Faltings. Constraint-based support for negotiation in collaborative design, *Artificial Intelligence in Engineering*, Vol. 14, No. 3, pp. 261-280, 2000.

[185] T. Laakko and M. Mantyla. Incremental constraint modeling in a feature modeling system, *Proceedings of EUROGRAPHICS'96*, J. Rossignac and F. Sillion (Eds.), 1996.

[186] W. Bouma, I. Fudos, C. Hoffmann, J.-Z. Cai, and R. Paige. A geometric constraint solver, *Computer-Aided Design*, Vol. 27, No. 6, pp. 487-501, 1995.

[187] E. Tsang. *Foundations of constraint satisfaction*, Academic Press, 1993.

[188] M. Sannella. The SkyBlue constraint solver, *Technical Report* 92-07-02, Department of Computer Science and Engineering, University of Washington, USA, 1992.

[189] G. A. Kramer. *Solving geometric constraint systems: a case study in kinematics*, The MIT Press, 1992.

[190] J. R. Rossignac. Constraints in constructive solid geometry, *Proceedings of the 1986 Workshop on Interactive 3D Graphics*, F. Crow and S. M. Pizer (Eds.), ACM Press, pp. 93-110, 1986.

[191] R. Light and D. Gossard. Modification of geometric models through variational geometry, *Computer-Aided Design*, Vol. 14, No. 4, pp. 209-214, 1982.

[192] G. Chen, Y.-S. Ma, G. Thimm, and S.-H. Tang. Knowledge-based reasoning in a unified feature modeling scheme, *Computer-Aided Design & Applications*, Vol. 2, No. 1-4, pp. 173-182, 2005.

[193] G. Pahl and W. Beitz. *Engineering design: a systematic approach*, 2nd Edition. London: Springer, 1996.

[194] W. Y. Zhang, S. B. Tor, and G. A. Britton. Automated functional design of engineering systems, *Journal of Intelligent Manufacturing*, Vol. 13, No. 2, pp. 119-133, 2002.

[195] Y.-M. Deng, G. A. Britton, and S. B. Tor. Constraint-based functional design verification for conceptual design, *Computer-Aided Design*, Vol. 32, No. 14, pp. 889-899, 2000.

[196] H.-P. Wang and J.-K. Li. *Computer-aided process planning*, Elsevier, 1991.

[197] Y.-Y. Zhang, S. C. Feng, X.-K. Wang, W.-S. Tian, and R.-R. Wu. Object-oriented manufacturing resource modeling for adaptive process planning, *International Journal of Production Research*, Vol. 37, No. 18, pp. 4179-4195, 1999.

[198] G. Thimm, G. A. Britton, and J.-T. Lin. Operation element-based efficiency for process plans and designs, *International Journal of Advanced Manufacturing Technology*, Vol. 24, No. 5-6, pp. 370-375, 2004.

[199] G. Halevi and R. D. Weill. *Principles of process planning: a logical approach*, Chapman & Hall, 1995.

[200] Spatial Corp., ACIS 3D geometric modeler, Version 7.0, 2001.

[201] MySQL AB, *MySQL Reference Manual for Version 5.0.1-alpha*, http://dev.mysql.com/doc/mysql

[202] R. E. Giachetti and M. I. Alvi. An object-oriented information model for manufacturability analysis of printed circuit board fabrication, *Computers in Industry*, Vol. 45, No. 2, pp. 177-196, 2001.

[203] H.-W. Law, H.-Y. Tam, A. H. S. Chan, and I. K. Hui. Object-oriented knowledge-based computer-aided process planning system for bare circuit boards manufacturing, *Computers in Industry*, Vol. 45, No. 2, pp. 137-153, 2001.

[204] A. J. C. Trappey, T.-H. Liu, and C.-T. Hwang. Using EXPRESS data modeling technique for PCB assembly analysis, *Computers in Industry*, Vol. 34, No. 1, pp. 111-123, 1997.

[205] Industrial automation systems and integration: product data representation and exchange: application protocol: Part 210 – Electronic printed circuit assembly, design and manufacture, *International Organization for Standardization (ISO)*, Geneva, Switzerland, 1994.

[206] Aberdeern group. The mechatronics system design benchmark report: Coordinating engineering disciplines, From http://www.ugs.com. 2006.

[207] D. Vossler and V. Dutt. Global trends and best practices in mechatronics product lifecycle management, From http://www.memagazine.org. 2005.

[208] P. Jackson. *Introduction to expert systems*, 3rd Edition, Addison-Wesley, 1999.

[209] Junhwan Kim and Soonhung Han. Encapsulation of geometric functions for ship structural CAD using a STEP database as native storage, *Computer-Aided Design*, Vol. 35, No. 13, pp. 1161-1170, 2003.

Appendix A. Change Propagation Algorithm

This appendix lists the pseudo code of the change propagation algorithm.

```
PROCEDURE Check_Local(x)
/* x is the initially modified variable. */
BEGIN
/* x_b is the back-up of x, x_n is the new value of x,
 * S_local-1 is the set of all the local variables affected by x,
 * S_local-2 is a set of affected local variables that associate to variables of other
 * applications,
 * mark1 indicates whether v_in violates any related constraints,
 * mark3 indicates whether x_n is locally accepted.
 */
  x_b←x; S_local-1←{}; S_local-2←{};
  mark1←false; mark3←false;   /* initialization */

  S_local-1←S_local-1+{x};

  FOR each v_i in S_local-1 DO
  BEGIN
    Search_JTMS(v_i);

    FOR each vm_i in S_driving DO
    BEGIN
    /* v_in is the new value of v_i, Evaluate( ) is a procedure to evaluate a constraint. */
      IF(Evaluate(v_in,vm_i,Cm_i)==violated) THEN
      BEGIN
        mark1←true; /*  v_in violates related constraints. */
        mark3←true; /*  x_n is not locally accepted. */
        Abandon(x);
      END
    END

    FOR each vs_i in S_driven DO
    BEGIN
      IF(Evaluate(v_in,vs_i,Cs_i)==violated) THEN
      BEGIN
        IF(vs_i is fixed) THEN
        BEGIN
          mark1←true;
          mark3←true;
          Abandon(x);
        END
        ELSE IF(vs_i has alternative values vs_in) THEN
        BEGIN
          mark2←false;  /* mark2 indicates whether Cs_i can be re-satisfied. */

          FOR each vs_in DO
          BEGIN
            IF(Evaluate(v_in,vs_in,Cs_i)==satisfied) THEN
            BEGIN
```

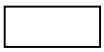
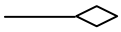
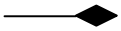
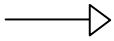
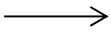
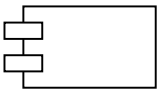
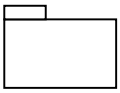
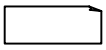
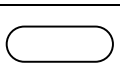
```
                vs_ib←vs_i;  /* vs_ib is the backup of vs_i. */
                vs_i←vs_in;
                mark2←true;        /* Cs_i is re-satisfied. */
                S_local-1←S_local-1+{vs_i};
              END
            END

            IF(mark2==false)       /* Cs_i cannot be re-satisfied. */
            BEGIN
              mark1←true;
              mark3←true;
              Abandon(x);
            END
          END
        END
      END

      IF(mark1==false) THEN   /* v_in does not violate any  related constraints. */
      BEGIN
        S_local-1←S_local-1−{v_i};

        IF v_i appears in DB THEN S_local-2←S_local-2+{v_i};
        /* i.e. stored in the association table in the database. */
      END
    END

    IF(S_local-1 is empty AND mark3==false) THEN
    /* x_n is locally accepted. */
    BEGIN
      x_n is locally accepted;

      FOR each v_i in S_local-2 DO Check_Global(v_i);
    END
END /* of Check_Local */

PROCEDURE Check_Global()
BEGIN
/* v_ib-asso is the back-up of v_i-asso, v_in-asso is the new value of v_i-asso. */
    Search_Database();

    FOR each v_i-asso in S_asso DO
    BEGIN
      v_ib-asso←v_i-asso;

      Solve(v_in,v_i-asso,C_i-asso);
      /* Solve( ) is a procedure to solve a constraint using v_in, the returned value is
       * v_in-asso.
      */
      v_i-asso←v_in-asso;

      mark4←false;       /* mark4 indicates whether x_n is globally accepted. */
```

```
    FOR each v_i-asso in S_asso DO
    BEGIN
      Check_Local(v_i-asso);

      IF(v_in-asso is not locally accepted) THEN mark4←true;
      /* x_n is not globally accepted. */

      ELSE IF(v_in-asso is locally accepted)
      THEN S_asso←S_asso−{v_i-asso};
    END

    IF(S_asso is empty AND mark4==false)
    THEN x_n is globally accepted;
    ELSE Abandon(x);
  END
END /* of Check_Global */

PROCEDURE Search_JTMS(x)
/* x is the modified variable */
BEGIN
/* S_driving is a set of variables vm_i (and its corresponding constraints Cm_i) which are the
 * antecedents of x in JTMS,
 * S_driven is a set of variables vs_i (and its corresponding constraints Cs_i) which are the
 * consequents of x in JTMS,
 * N is the set of all nodes n_i in JTMS. Each node n_i corresponds to one and only one
 * variable v_i.
 */
  FOR each n_i in N DO
  BEGIN
    IF n_i is the antecedent of x
    THEN S_driving←S_driving+{<v_i, Cm_i>};
    ELSE IF n_i is the consequents of x
    THEN S_driven←S_driven+{<v_i, Cs_i>};
  END
END /* of Search_JTMS */

PROCEDURE Search_Database()
BEGIN
/* S_asso is a set of variables v_i-asso (of other applications) that associates to the variables v_i in
 * S_local-2 via constraints C_i-asso. */
   S_asso←{};

  FOR each v_i in S_local-2 DO S_asso←S_asso+{<v_i-asso, C_i-asso>};
END /* of Search_Database */

PROCEDURE Abandon(x)
BEGIN
  x←x_b;
  FOR each v_i in S_local-1 DO v_i←v_ib;
  FOR each v_i in S_local-2 DO v_i←v_ib;
  FOR each v_i-asso in S_asso DO v_i-asso←v_ib-asso;
```

```
    S_local-1 ← { };
    S_local-2 ← { };
    S_asso ← { };
END /* of Abandon */
```

Appendix B. List of UML Notations

The following UML notations are used in Figures 3.1, 4.4, 5.8, 5.9, 7.2, 7.3, and 7.4.

	class
	interface
	dependency relationship
	aggregation relationship
	composition relationship
	generalization relationship
	association relationship
	navigation across an association relationship in a class diagram
« …… »	stereotype
	component
	package
	association class
	note
	start state
	stop state
	action state or activity state
	branch
[……]	guard expression