Next Generation Concurrent Engineering - M. Sobolewski & P. Ghodous (eds)
© 2005 ISPE, Inc., ISBN 0-9768246-0-4

A Unified Feature Modeling Scheme for Multi-applications in PLM

G. Chen[a], Y.-S. Ma[a,*], X.G. Ming[b], G. Thimm[a], Stephen S.G. Lee[a], L.-P. Khoo[a], S.-H.Tang[a], W. F. Lu[b]

[a] *School of Mechanical and Aerospace Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798. * Corresponding author, Email: mysma@ntu.edu.sg, Tel: 65-6790 5913*

[b] *Singapore Institute of Manufacturing Technology, 71 Nanyang Drive, Singapore 638075. Email: hxgming@ieee.org, Tel: 65-6793 8983*

ABSTRACT: Currently, collaborations in Product Lifecycle Management (PLM) become a major trend. However, different product lifecycle phases are interrelated. Although collaborative product development approach has been suggested to find a balance among usually conflicted requirements of these phases, but very few in-depth research works have addressed the inherent sequential and iterative natures of the product development processes coherently. Due to the inter- or intra-phase relationships, a chain of changes is very likely to occur. Modeling and maintaining these relations are hence important in collaborative engineering to evolve the state of the whole system in a stable and consistent manner. This paper elaborates a feature-based, object-oriented and unified approach for the information sharing among multiple applications of PLM collaboration.

Keywords: Product Lifecycle Management, Feature Unification, Collaborative Product Development, Collaborative Manufacturing.

1 INTRODUCTION

The whole product lifecycle include many phases, such as conceptual design, detailed design, analysis, manufacturing, assembling, etc. These phases are interrelated. Even though different phases may have different requirements on the semantics, product structures and specifications, however, different phases connect to each other on the basis of the common detailed-design models. One functional requirement may have several design solutions while one specific detailed design may be produced using different process plans. To realize better product quality with less cost in a shorter development cycle, as an extension of concurrent engineering, PLM collaboration is proposed as a methodology to consider requirements from different phases comprehensively for reaching an optimized overall product performance.

Besides the interrelations, the product development processes are inherently sequential and iterative although overlaps exist in some extent. Downstream phases usually need at least a partial solution from an upper stream phase as an input. Downstream phases might also ask for changes of one or more upper stream phases to fulfill its constraints (Thimm, G. et al. 2004a, and b). Any change in a single phase might invoke a chain of changes in its own or other phases (Eastman, C.M. 1996). Current applications are usually standalone. Any change of the input makes the whole execution process restart from scratch. Managing the inter- or intra-phase relations can make change propagation more efficiently and hence ensure a consistent product information model in concurrent engineering. The current research assumes these applications (corresponding to different phases) have a centralized information source, i.e. they are integrated systems. However, for distributed and usually heterogeneous systems where there is not any centralized database; the implementations of information consistency control are different and more difficult although the rationale is similar (Li, W.D. et al. 2004).

Some product lifecycle phases, such as machining planning or assembly planning, are more geometry-related than others. Features, which can associate non-geometric information with geometric entities are suitable to be used as information units in these phases for the following reasons: (1) As mentioned above, these geometry-related phases connect to each other on the basis of the common product structures and specifications. Different features can communicate with each other via direct non-geometric attribute/constraint relations (e.g. specifications) or via indirect common geometry- based relations (e.g. structures) and hence support more comprehensive communications within a single phase or between phases; (2) In each individual phase, feature model can be used as an intermediate layer vertically between the knowledge-oriented reasoning processes (mainly analysis and manipulation on non-geometric entities) and the procedural engineering processes (manipulation on and interactions with geometry models, which consist of geometric entities). (3) A unified feature modeling scheme was proposed by the authors (Chen, G. et al. 2004) to use features as the intermediate information layer for change propagation and information consistency control in the product development processes. In this scheme, a feature is generically defined as a combination of geometric entities, non-geometric attributes as well as explicitly defined inter- or intra-feature relations. Using object-oriented terms, a feature object also keeps references to other relevant information entities, which include knowledge, geometry or other features. Therefore, the unifications can be achieved between different phases as well as in each individual phase.

This paper elaborates the definitions of the entity classes and the relation classes used in a unified feature-modeling scheme. The ultimate goal of the proposed scheme is for information consistency control. This research is the continued effort following the previous paper (Chen, G. et al. 2004). This paper is organized as follows: section 2 reviews some past related research; object-oriented definitions of different application features are given in section 3; section 4 discusses definitions of relation classes; the algorithms for change propagation are briefly described in section 5; a case study is described in section 6; finally, conclusions are given.

2 RELATED RESEARCH

There is a school of thought based on multiple-view feature modeling. It is believed that a specific application feature model is extracted from, and regarded as a view of, the whole product information model. De Martino et al. (1998) used common faces to propagate modifications from one view to another. Bronsvoort & Noort (2004) proposed a multiple-view feature modeling framework which integrates the conceptual design, detail design, assembly design and manufacturing planning views. The non-geometric relations were discussed between the conceptual design view and other views. Subramani & Gurumoorthy (in press) used volumetric intersections to propagate geometric feature modifications. In general, semantic relations between different feature models have not been fully explored yet.

Regarding to the phase of conceptual design, Gui & Mantyla (1994) mentioned the importance of geometric modeling in the conceptual design stage. The mapping from the required functions to the product structures and the conversion from function relations to spatial relations were discussed. Ranta et al. (1996) pointed out the possible correspondences between functional entities and geometric constraints. Gorti et al. (1998) proposed an object-oriented formalism to represent a design artifact that includes the information about the required functions, behaviors and the resulted forms.

As to CAE analysis, Deng et al. (2002) developed a feature-based CAD-CAE integration model. A set of CAE features, which regroup CAD features with the addition of analysis-specific attributes (e.g. suppressibility) and user-defined constraints, is used as input for CAE analysis. Lee (in press) proposed a non-manifold data structure to cover different geometric modeling requirements of CAD and CAE systems. Different geometric abstraction levels between CAD and CAE features are highlighted.

In the phase of assembly planning, besides the function-oriented assembly design, assembly planning focuses more on the product assembling process, such as the assembling methods, assembly sequence, etc. Holland & Bronsvoort (2000) discussed the possible usages of feature concept during assembly planning stage for the purpose of stability analysis, motion planning and sequence analysis. Kim et al. (2004) analyzed the required spatial relations between joined parts based on the specified assembling methods.

When considering machining planning, although feature technology originated from the purpose of integrating design and process planning, most of the early-proposed machining features are rigid geometric patterns used for feature recognition. Khoshnevis et al. (1999) used a feature completion module to convert design features into machining features, which are suitable for process planning. Stage et al. (1999) proposed a machining resource-based and objective driven method for embedding manufacturing knowledge into a machining feature generation process.

There exist many different intra- or inter-phase relations. Ma & Tong (2003, 2004) revealed the importance of using the associative nature of features in a design process. An associative feature concept was proposed to model associated geometric feature entities. It is highlighted that an ideal data structure of a feature definition must be flexible and self-contained. The consistency of relations has to be kept by using the essential geometric entities and their relations while the different representations, that are geometric as well as non-geometric, are generated and managed. In the unified feature modeling scheme, this associative feature concept was generalized and extended to include broader aspects of relations among PLM phases. Kusiak & Wang (1995) proposed a general dependency network to manage relations between design variables for modification propagation. Eastman (1996) used the precedence relations between application operations to manage the validity of product information. Park & Cutkosky (1999) generalized three basic types of relationship in a design process: precedence relations among tasks, constraint relations among design elements and different abstraction levels. The above reviewed researches illustrate that: (1) Features can be modeled as a set of generic geometry related object structures with levels of abstraction of properties and methods such that they can be used in different geometry-related phase as information units; (2) In real applications, a feature is an application-specific object instance which relates a group of geometric entities by the means of object polymorphism. Non-geometric attributes are used in the application's reasoning processes; (3) Relationship management is crucial to maintain the validity of a specific feature model as well as the whole product information model.

3 ENTITY CLASS DEFINITIONS IN THE UNIFIED FEATURE MODELING SCHEME

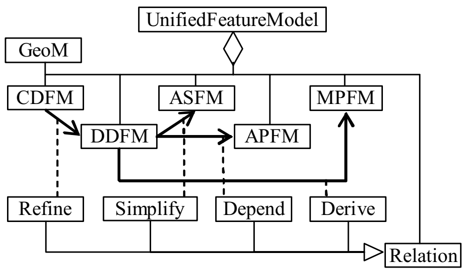

Figure 1. Unified feature model

A UML diagram (Fig. 1) is used to roughly illustrate the unified feature model concept. In the figure, GeoM represents geometry model while CDFM, DDFM, ASFM, APFM and MPFM represent conceptual design, detailed design, CAE analysis, assembly planning and machining planning feature model respectively. It is assumed that there is a centralized information source (the *UnifiedFeatureModel* class) for different phases. This object maintains a list of application feature models (*AppFeatModel* objects) as well as the common product geometry model (*GeoModel* object). The *UnifiedFeatureModel* object also keeps a list of *Relation* objects. A *Relation* object is initiated whenever a relation is established between different applications. Each *Relation* object has a list of controlled variables, which belong to different applications and might be of different types, e.g. *Feature*, *Attribute* or *Geometry*, etc. The references of all the involved variables are also recorded in the Uni*fiedFeatureModel* object, i.e. each application registers its relevant information in the central information source. With these references, the responsibilities of the *UnifiedFeatureModel* class (realized through its member functions) include querying application models for values of variables as well as propagating modifications to relevant application feature models. Some base classes are also defined in this level, such as *FeatureModel*, *UnifiedFeature*, *Attribute*, *Constraint*, etc. For each application feature model, the corresponding application specific subclasses, such as *AppFeatModel*, *AppFeature*, etc. are derived from, and hence inherit generic attributes and methods of, these base classes. Fig. 2 gives a UML class diagram where the unified feature concept is illustrated. In the figure, CDFeat, DDFeat, ASFeat, APFeat and MPFeat represent conceptual design, detailed design, analysis, assembly planning and machining planning feature classes respectively.

Each application model is an aggregation of three sub-models of information: *KnowledgeBase*, *AppFeatureModel* and *AppGeoModel*. Fig. 3 is the UML diagram that shows the structure of an application information model. KM, FM and GM represent knowledge-based semantic model, application feature model and application specific geometry model respectively.

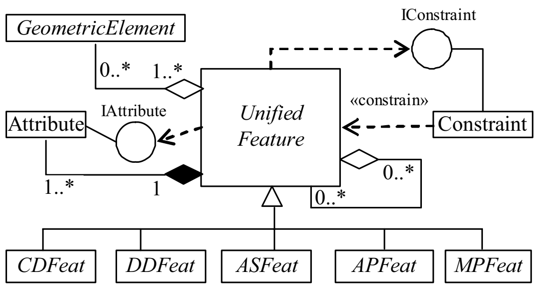

Figure 2. Unified feature definition

The *KnowledgeBase* class is responsible for simulating human engineers' reasoning process and recording their heuristics (rules). The application feature model is a hierarchical model which is divided into several levels, such as assembly, component, feature, etc. The assembly or component concept is defined here as a functionally independent unit, such that they are corresponding to a product subfunction or a machining process. They are not necessarily physically realizable and might be non-manifold geometry. In each level, a local interaction network is weaved by shared attributes and common constraints. Attribute and constraint objects can be initiated in different levels, such as assembly or feature constraints. Each attribute or constraint object maintains a list of used entity pointers (rule pattern, assembly, feature, etc.) that refer to it. This list is used for change propagation and feature/rule validity check when modifying application specific variables.

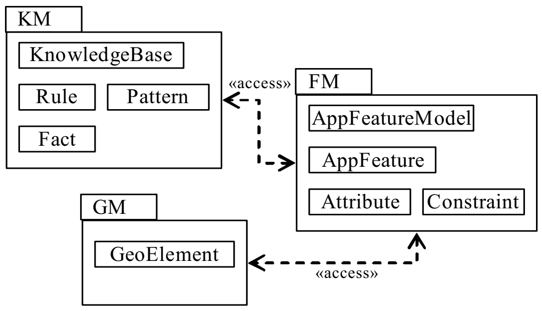

Figure 3. An application information model

Each specific feature class (*AppFeature*) is a relatively independent information unit in the reasoning process of its owning application, such as functional design, assembly planning or machining planning. Each feature class has some accessible attributes and executable methods, i.e. clearly defined interfaces that can be invoked locally or remotely. The application geometry models (*AppGeoModel*) might be either 2-manifold or non-manifold according to application specific modeling requirements (Sriram, R.D. et al. 1995). For example, the geometry models of the conceptual design, CAE analysis and machining planning are non-manifold. They might also have mixed dimensionalities.

3.1 *Conceptual design feature model*

Generally, the tasks during the conceptual design stage include function to behavior mapping and behavior to structure mapping. They are not one-to-one mappings. The mapping or selection heuristics are recorded in the corresponding knowledge bases. Mechanical functions and behaviors of a product are usually expressed as the interactions between pairs of critical faces in the form of relative positions, orientations, fit relations or relative motions as well as other critical dimensions/specifications. Each conceptual design feature instance corresponds to a relatively independent sub-function. Critical faces are conceptual features' geometric entities and other specifications are features' attributes or constraints. Each conceptual feature instance records its owning functions as well as the behaviors it participates in. Different feature instances interact to each other through their interrelated functions or behaviors. The result of conceptual design can be described as a skeleton of the product, which consists of abstracted critical faces, constraints, and other specifications (functions and behaviors).

3.2 *Detailed design feature model*

In this phase, according to the skeleton specified during the conceptual design phase, the complete product geometry and specifications are developed gradually. Traditional form features are applied mainly in this domain. In the proposed unified feature modeling scheme, when constructing a detailed design feature model, the specified conceptual design features should be transferred into and associated with the corresponding detailed design features. Relations between the critical faces of a conceptual design feature instance are usually converted into multi-level relations across sub-assemblies, parts, features in the detailed design feature model, such as the relative position, orientation, motion, connection and fit relations. The detailed design can be seen as a refinement process, where detailed design feature model is enriched gradually by the embodiment from critical faces into parts, sub-assemblies. Each part encloses a set of detailed form features; and each (sub-) assembly consists of a set of parts with mating conditions. These corresponding reference or refinement relations are registered in the *UnifiedFeatureModel* object.

3.3 *Analysis feature model*

In general, the feature model for CAE analysis is a simplification of the corresponding detailed design feature model. Some minor design details might be suppressed. Some geometric entities in the detailed design feature model might degenerate into lower dimensionality entities. For example, during the cooling system design for an injection mold, a solid representation of a cooling channel might be simplified as a cooling line during cooling effect analysis. According to its type, dimensions or user specifications, a detailed design feature might be suppressed or simplified. These simplification heuristics or criteria are kept in the corresponding knowledge bases. The analysis specific attributes, such as suppressibility or material, are included in each analysis feature instances. The simplification and analysis methods are defined in analysis feature classes. The correspondence between the detailed design features and the analysis features are recorded in the *UnifiedFeatureModel* object.

3.4 *Assembly planning feature model*

The assembly planning process focuses on evaluating the feasibility of an assembling method, motion planning and assembly sequence planning. The part geometry, material, position, orientation, fit or connection relations specified during the detailed design phase are used as input to the assembly planning process. An assembly planning feature may represent a group of geometric entities which are significant during assembly stability analysis, reliability analysis, motion analysis, etc. Similarly, relevant heuristics are stored in the knowledge bases. The correspondence between the detailed design features and the assembly planning features are recorded in the *UnifiedFeatureModel* object.

3.5 *Machining planning feature model*

Machining planning feature model is derived from the detailed design feature model according to design specifications, available manufacturing resources and user-specified objectives, such as cost or time. A single face of a detailed design feature might be associated to a set of machining features (including intermediate ones), which are derived from the detailed design face according to the machining plan.

The geometry entities of a machining plan feature might include one or a set of faces in the final or intermediate product geometry. The machining plan features interact with other relevant classes, such as machine, tool, setup and machining knowledge, to generate a feasible and optimized machining plan. If necessary, those feature faces can be converted into volumes removed during a machining process for cutting time/cost calculations. The machining specific attributes, such as used machine, tools, setups, are defined in each machining plan feature class. Similarly, the correspondence between the original detailed design features and the machining plan features are stored in the *UnifiedFeatureModel* object.

4 RELATIONS IN THE UNIFIED FEATURE MODELING SCHEME

First, there are four types of inter-phase relations between different applications in the unified feature-modeling scheme: simplification, refinement, derivation and dependency. Simplification can be exemplified with relations between the detailed design geometry and the geometry for CAE analysis. Refinement can be typically illustrated by the relations between the conceptual design geometry and the detailed design geometry. Derivation represents the kind of relations between a detailed design geometric surface and the geometry of the corresponding intermediate surfaces in its related machining features. Dependency is the most common relation between application feature models. It may represent precedence or consequence relations between applications. It may also represent value sharing relationship, i.e. variables in different applications must have the same value. Second, as analyzed in (Chen, G. et al., accepted), intra-phase relations within an individual application information model, can be divided into three levels: relations between the knowledge base and the feature model, intra- or inter-feature relations, and relations between the feature model and the geometry model. They can be classified into three types: dependency, referring and binding. Dependency means that the existence and validity of a knowledge fact or a feature entity depend on the existence and validity of other feature entities or knowledge facts. Referring means the associative relations between a feature and its used geometric elements (in the *AppGeoModel*) via pointers. Binding relates a feature attribute to the corresponding feature geometric entities. These relations are used to construct a dependency network for modification propagation and information consistency control.

5 CHANGE PROPAGATION

The algorithm for change propagation in the unified feature modeling scheme can be roughly divided into three steps: (1) When the value of a feature entity in an individual application feature model is modified, other affected feature entities are found via the common attributes, constraints or shared geometric elements; (2) The relevant rules in the knowledge base of this application are checked for the validity of the original design intents related to this modified variable. This checking might result in other related rules be fired or retracted. In consequence, related facts and hence feature entities might be changed as well. These first two steps are used to recover the information consistency locally, i.e. in an individual application model; (3) If the modified variable is registered in the *UnifiedFeatureModel*, the *Relation* objects are used to find other affected applications. Another round of local validation and modification process begins by executing step (1) and (2) repeatedly until a solution is found (i.e. all specified constraints are satisfied) or unfulfilled constraints are finally reported. To prevent infinite process loops, priorities of constraints are specified. Heuristic rules should used to enable dynamic adjustment of the priorities of constraints to prevent exhaustive search in a downstream application. For example, a strict tolerance specification may make the manufacturing cost too high to be reasonable. Rules should be applied to check if the functional requirements require such strict tolerances.

6 CASE STUDY

In this section, cooling system design in an injection mold assembly is used as a case to illustrate the advantages of the proposed unified feature modeling concept (Fig. 4). The structures and specifications of a cooling system include circuit type, size, number, position, orientation, etc. The decision on these design variables is influenced by the consideration of its functional aspect (cooling effect), assembly aspect (interference with other mold components or plastic part) and manufacturing aspect (drilling direction, length and number of drilling operations). In each aspect, the cooling system has different geometric representations. For example, in conceptual design phase, cooling circuits might be preferred to find possible design candidates. The analysis of cooling effect may need a solid or mesh representation. Interference checking with other components or manufacturing planning phase requires for a solid representation. Different channels of the same cooling circuit must be connected. They are associated by the geometric constraints. Member circuits of the same cooling system are constrained by the functional requirement, i.e. they are combined together to reach a required cooling effect. Hence, a change of a single cooling channel might invoke a chain of changes of the remaining cooling components. Without an efficient and effective integrity control mechanism, managing these relations and hence keeping information consistency will be time-consuming and error-prone. Interesting readers may refer to (Ma, Y.-S. & Tong, T. 2003) for details of implementation.

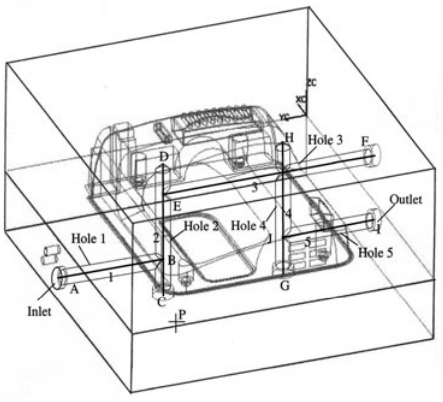

Figure 4. A typical case modeled with unified feature definition

7 CONCLUSION

Because the different phases of the PLM process are interrelated and the product development process is inherently sequential and iterative, keeping information consistency is crucial but also very complicated. This paper presents a unified feature modeling scheme in which features are used as an intermediate information layer to glue the declarative knowledge models and the procedural data (e.g. geometry) models. The basic idea is to regard the whole product development process as well as the whole product information model as a huge dependency network. Modifying a single node (a variable) in this network may have different scopes of influences. Different application feature models communicate with each other through direct feature relations as well as indirect geometry-based relations. Several entity classes and relationship types are generalized. The proposed unified feature-modeling scheme provides a framework to address the dependencies comprehensively and systematically.

REFERENCES

Bronsvoort, W.F. 2004. Multiple-view feature modeling for integral product development. *Computer-Aided Design* 36(10): 929-946.

Chen, G., Ma, Y.-S., Thimm, G. & Tang, S.-H. 2004. Unified feature modeling scheme for the integration of CAD and CAx. *Computer-Aided Design & Applications* 1(1-4): 595-602.

Chen, G., Ma, Y.-S., Thimm, G. & Tang, S.-H., Knowledge-base reasoning in a unified feature modeling scheme. Accepted by *Computer-Aided Design & Applications*.

de Martino, T., Falcidieno, B. & Habinger, S. 1998. Design and engineering process integration through a multiple view intermediate modeler in a distributed object-oriented system environment. *Computer-Aided Design* 30(6): 437-452.

Deng, Y.-M., Britton, G.A., Lam, Y.C., Tor, S.B. & Ma, Y.-S. 2002. Feature-based CAD-CAE integration model for injection-moulded product design. *International Journal of Production Research* 40(15): 3737-3750.

Eastman, C.M. 1996. Managing integrity in design information flows. *Computer-Aided Design* 28(6/7): 551-565.

Gorti, S.R., Gupta, A., Kim, G.J., Sriram, R.D. & Wong, A. 1998. An object-oriented representation for product and design processes. *Computer-Aided Design* 30(7): 489-501.

Gui, J.-K. & Mantyla, M. 1994. Functional understanding of assembly modeling. *Computer-Aided Design* 26(6): 435-451.

Khoshnevis, B., Sormaz, D.N. & Park, J.Y. 1999. An integrated process planning system using feature reasoning and space search-based optimization. *IIE Transactions* 31(7): 597-616.

Kim, K.-Y., Wang, Y., Muogboh, O.S. & Nnaji, B.O. 2004. Design formalism for collaborative assembly design. *Computer-Aided Design* 36(9): 849-871.

Kusiak, A. & Wang, J. 1995. Dependency analysis in constraint negotiation. *IEEE Transactions on Systems, Man, and Cybernetics* 25(9):1301-1313.

Lee, S.-H., in press. A CAD-CAE integration approach using feature-based multi-resolution and multi-abstraction modeling techniques. *Computer-Aided Design*.

Li, W.-D., Ong, S.K., Fuh, J.Y.H., Wong, Y.S., Lu, Y.Q. & Nee, A.Y.C. 2004. Feature-based design in a distributed and collaborative environment. *Computer-Aided Design* 36(9): 775-797.

Ma, Y.-S. & Tong, T. 2003. Associative feature modeling for concurrent engineering integration. *Computers in Industry* 51(1): 51-71.

Ma, Y.-S. & Tong, T. 2004. An object-oriented design tool for associative cooling channels in plastic-injection moulds. *International Journal of Advanced Manufacturing Technology*, 23(1-2): 79-86.

Park, H. & Cutkosky, M.R. 1999. Framework for modeling dependencies in collaborative engineering processes. *Research in Engineering Design* 11(2): 84-102.

Ranta, M., Mantyla, M., Umeda, Y. & Tomiyama, T. 1996. Integration of functional and feature-based product modeling – the IMS/GNOSIS experience. *Computer-Aided Design* 28(5): 371-381.

Sriram, R.D., Wong, A. & He, L.-X. 1995. GNOMES: an object-oriented nonmanifold geometric engine. *Computer-Aided Design* 27(11): 853-868.

Stage, R., Roberts, C. & Henderson, M. 1999. Generating resource based flexible form manufacturing features through objective driven clustering. *Computer-Aided Design* 31(2): 119-130.

Subramani, S. & Gurumoorthy, B., in press. Maintaining associativity between form feature models. *Computer-Aided Design*.

Thimm, G., Britton, G.A. & Fok, S.C. 2004. A graph theoretic approach linking design dimensioning and process planning Part 1: Designing to process planning. *International Journal of Advanced Manufacturing Technology* 24(3-4): 261-271.

Thimm, G., Britton, G.A. & Fok, S.C. 2004. A graph theoretic approach linking design dimensioning and process planning Part 2: Design heuristics for rotational parts. *International Journal of Advanced Manufacturing Technology* 24(3-4): 272-278.

van Holland, W. & Bronsvoort, W.F. 2000. Assembly features in modeling and planning. *Robotics and Computer Integrated Manufacturing* 16(4): 277-294.