Available online at www.sciencedirect.com
ScienceDirect
Computers in Industry 59 (2008) 110–118
COMPUTERS IN INDUSTRY
www.elsevier.com/locate/compind

Change propagation algorithm in a unified feature modeling scheme

Yongsheng Ma [1,*], Gang Chen, Georg Thimm

School of MAE, Nanyang Technological University, Singapore 639798, Singapore

Available online 10 July 2007

Abstract

Product lifecycle stages are inter-related and mutually constraining. Due to the sequential nature of the product development processes, some constraints or conflicts may emerge in a later stage and require modifications to the decisions made in earlier stages. The iterations between stages are hence unavoidable and must be managed carefully to maintain the consistency, integrity and validity of product information models. Due to the inter- or intra-stage relations, a chain of changes is very likely to occur as the consequence of an initial change. Modeling and maintaining these relations are important in collaborative engineering to evolve the state of the whole product model in a consistent manner. This paper introduces a new method of modeling associative engineering relations in a unified feature modeling scheme and elaborates a change propagation algorithm for the information consistency control among multiple applications of product lifecycle stages. The algorithm is established on a JTMS-based dependency network. Two case studies are used to illustrate the proposed dependency network and change propagation algorithm.

Keywords: Product lifecycle modeling; Unified feature; Data consistency; Change propagation

1. Introduction

A product lifecycle can be divided into several stages; they are design, manufacturing, marketing, use, and recycling. Along with them are different engineering processes, such as conceptual design, detail design, CAE analysis, process planning, machining, assembly, and so on. A later stage needs the results of the earlier stages as input. At the same time, the later stage influences decisions made in the earlier stages [1,2]. One stage may also require modifications to the decisions made in other stages. In other words, these stages are inter-related and mutually constraining. Therefore, any modification in one stage can invoke a chain of subsequent intra- and inter-stage checking and modifications [3]. Modeling and managing the inter- or intra-stage relations can make change propagation more efficient and ensure a consistent product information model in concurrent engineering.

Incompatible data structures and lack of inter-stage associations are two major obstacles to realize data sharing and consistency control among lifecycle stages. Features, which can associate non-geometric information with geometric entities, are suitable to be used as information units to overcome these two obstacles. A unified feature modeling scheme was proposed by the authors [4,5] to use features as an intermediate information layer for change propagation and information consistency control in the product development processes. This scheme is characterized by a unified feature definition, data association mechanisms as well as the algorithms for propagating modifications. Two major extensions to the traditional feature technology are the embedded knowledge-based reasoning [6] and the unified, cellular topology based feature modeling mechanisms.

This paper elaborates a change propagation algorithm used in the unified feature modeling scheme. The paper is organized as follows: Section 2 reviews some past related research; the unified feature modeling scheme is introduced in Section 3 to make the paper self-contained; Section 4 analyzes the workflow in three lifecycle stages to identify the intra- and inter-stage relations; Section 5 presents the algorithms for change propagation; two case studies are described in Section 6; finally, conclusions are given.

2. Related research

Multiple-view feature modeling approach regards a particular application feature model as a view of the whole product information model. De Martino et al. [7] used common faces to

* Corresponding author. Tel.: +1 780 492 4443; fax: +1 780 492 2200.
E-mail address: yongsheng.ma@ualberta.ca (Y. Ma).
URL: http://www.ualberta.ca/~youngshen
[1] Present address: Department of Mechanical Engineering, University of Alberta, Canada.

0166-3615/$ – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.compind.2007.06.006

propagate modifications from one view to another. Subramani and Gurumoorthy [8] used volumetric intersections to propagate geometric feature modifications. They only explored the geometric relations between feature models. Bronsvoort and Noort [9] 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 identified between the conceptual design view and other views. In general, semantic relations between different feature models have not been fully explored yet.

Regarding to the stage of conceptual design, Schulte et al. [10] proposed using functional features to support the conceptual design activities. Using functional features to represent design intent via their relations to the physical effects was described. Henderson [11] proposed a two-realm framework to link conceptual and detail design. Product definition units were used to represent design functionalities. Gui and Mantyla [12] 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. [13] pointed out the possible correspondences between functional requirements and geometric constraints. Gorti et al. [14] proposed an object-oriented formalism to represent a design artifact that includes the information about the required functions, behaviors and the resulted forms. Chakrabarti and Bligh [15] analyzed the three major functional reasoning approaches for conceptual design. They concluded that a practical functional reasoning model must support designs of any nature and in any levels of detail. Qin et al. [16] proposed using behavioral modeling to support conceptual design process. Aleixos et al. [17] explored using reference datum, axes, curves, surfaces, coordinate systems, and inter-part relations to model and control the crucial product characteristics during the embodiment design stage. Han and Lee [18] proposed a case-based reasoning method through indexing, retrieving and adapting virtual function generators for mechanism design.

In the stage 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. Van Holland and Bronsvoort [19] discussed the using of feature concept during assembly planning stage for the purpose of stability analysis, motion planning and sequence analysis. Gottipolu and Ghosh [20] generate feasible assembly sequences through analyzing the geometric and mobility constraints from the detail design. Kim et al. [21] analyzed the required spatial relations between joined parts based on the specified assembling methods.

When considering process planning, Khoshnevis et al. [22] used a feature completion module to convert design features into machining features, which are suitable for process planning. Stage et al. [23] proposed a machining resource-based, objective driven and hence flexible method for embedding manufacturing knowledge into a machining feature generation process. Chu and Gadh [24], Kumar et al. [25], and Subrahmanyam [26] discussed that fixturing features should be generated from the considerations of minimizing set-ups, preventing interferences and providing sufficient clamping forces and moments. These research works demonstrate that feature is not a pure geometric concept, feature semantics must be considered when defining and using application features.

There exist many different intra- or inter-stage relations. Ma and Tong [27,28] revealed the importance of using and maintaining the associative nature of features during the design processes. An associative feature concept was proposed to model the associated geometric entities via a generic feature class. It was highlighted that features should be self-contained and flexible. They should have built-in associative links, different representation forms and self-validation methods. The consistency of relations has to be managed while the geometric representations pertaining to various lifecycle stages are evolved. In the unified feature modeling scheme, this associative feature concept was extended to include non-geometric relations among lifecycle stages. Kusiak and Wang [29] proposed a general dependency network to manage relations between design variables for modification propagation. Eastman [3] used the precedence relations between application operations to manage the validity of product information. Park and Cutkosky [30] 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 research illustrates that:

(1) Each product lifecycle stage includes geometric and non-geometric data. Both of them must be represented in the product information model.
(2) Features can be modeled as a set of generic geometry-related objects with levels of abstraction of properties and methods such that they can be used as information units in different geometry-related stages.
(3) 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.
(4) Product lifecycle stages are inherently related. They must be managed together to maintain the information consistency. Relationship management is crucial to maintain the validity of a specific feature model as well as the whole product information model.

3. Unified feature modeling scheme

In the unified feature modeling scheme, a unified feature is generically defined as a combination of geometric references, non-geometric attributes as well as explicitly defined inter- or intra-feature relations. In Fig. 1, the unified feature definition is shown using a UML class diagram [31]. A unified feature object keeps references to other relevant information entities, which include knowledge (via attributes), geometry or other features. Application features are sub-classes of the unified feature class and hence inherit the generic characteristics and methods defined in the unified feature class.

As shown in Fig. 2, each application corresponding to a particular lifecycle stage consists of four modules,

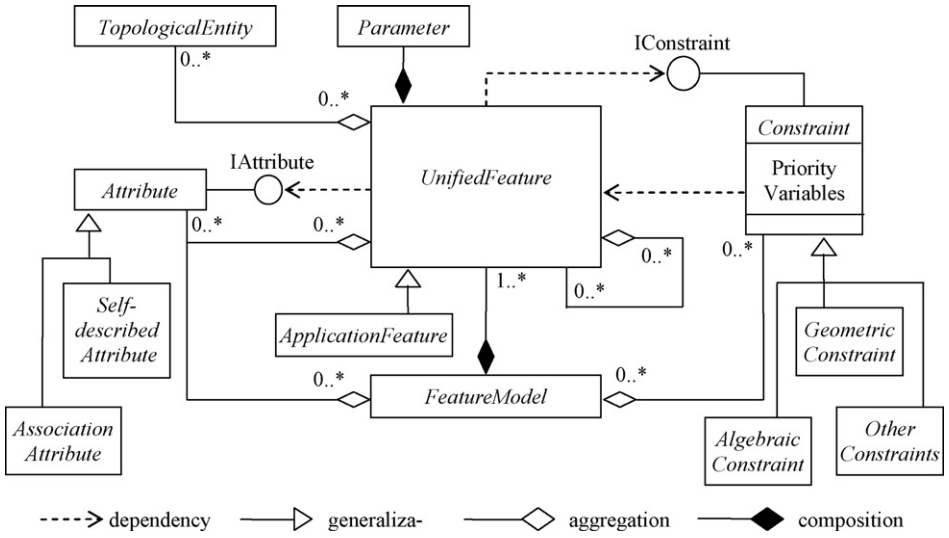


Fig. 1. Unified feature definition.

knowledge-based semantic module, application feature module, unified feature module, and geometric module. In the present research, a rule-based expert system is used to provide the knowledge-based reasoning capability. The knowledge-based semantic module communicates with the application feature module for invoking feature methods or updating a feature model. The application feature module uses the methods defined in the unified feature module, which include manipulating attributes, accessing geometric module, and solving constraints.

Based on these fundamental mechanisms, unified features can be used to associate 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).

Furthermore, the proposed unified features can also be used to associate lifecycle stages via two ways.

Firstly, indirect common geometry based associations can be established between stages. In the unified feature modeling scheme, application feature models use a common unified cellular model. The multi-dimensional, non-manifold, unified cellular model integrates geometries from all involved applications. The owning feature attributes of each cell are used to establish the common geometry based associations and to propagate geometric modifications across features.

Secondly, direct non-geometric constraint based associations can be established between stages. As shown in Fig. 3, product lifecycle stages are inherently related and mutually constraining. In the figure, GeoM represents geometry model while CDFM, DDFM, ASFM, APFM and PPFM represent

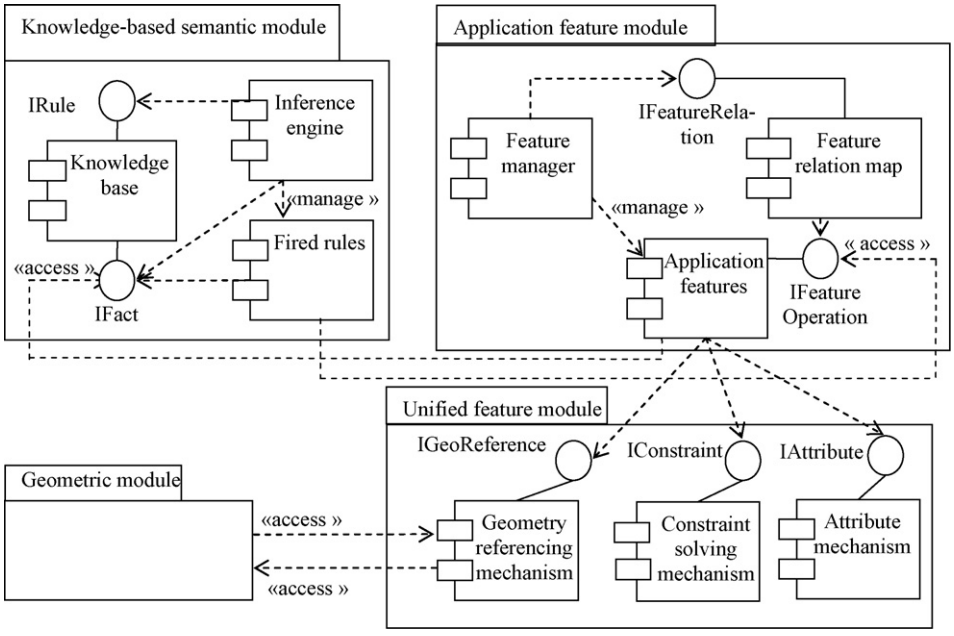


Fig. 2. Intra-stage associations in the unified feature modeling scheme.

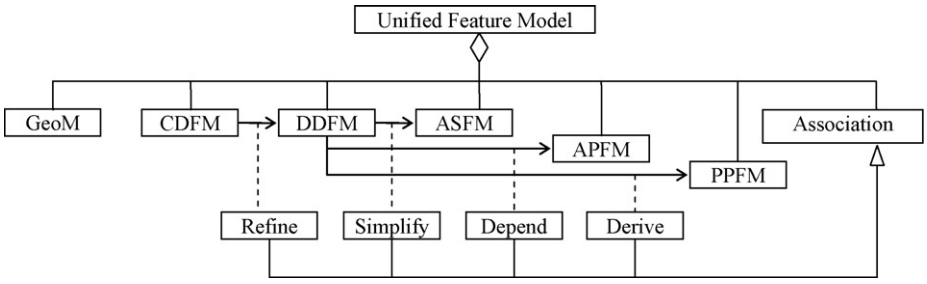


Fig. 3. Unified feature model.

conceptual design, detail design, CAE analysis, assembly planning and process planning feature model, respectively. The *Unified Feature Model* object keeps a list of *Association* objects, which are stored in a relational database. An *Association* object is initiated whenever an association is established between different applications. Each *Association* object has a list of controlled variables, which may belong to different applications and may be of different types, e.g. *Feature*, *Attribute*, *Fact*, or *Geometry*, etc. The references of all the involved variables are also recorded in the *Unified Feature Model* object, i.e. each application registers its relevant data in the central database. Different types of associations exist between particular stages. For example, detail design is a refinement of the conceptual design while the CAE analysis model is usually a simplification of the detail design.

With these references, the responsibilities of the *Unified Feature Model* 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 (Fig. 1), such as *Feature Model*, *Unified Feature*, *Attribute*, *Constraint*, *KnowledgeBase*, 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.

The *KnowledgeBase* class is responsible for emulating human engineers' reasoning process and recording their heuristics (as rules). The application feature model is a hierarchical model which is divided into several levels, such as assembly, component, feature, etc. To accommodate the requirements of the conceptual design stage, the assembly or component concept is defined here as a functionally independent unit, such that they correspond to product sub-functions. They are not necessarily physically realizable and may have non-manifold geometry. In each level, a local dependency network is weaved by shared attributes and common constraints. Attribute and constraint objects can be initiated in each level, such as assembly or feature constraints. Each attribute or constraint object maintains a list of used entity references (rule patterns, assemblies, features, etc.) that refer to it. This list is used for change propagation and feature/rule validity check when application variables are modified.

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 process 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 may be either two-manifold or non-manifold according to application specific modeling requirements. For example, the geometry models of the conceptual design may be non-manifold. They may also have mixed dimensionalities.

4. Application feature models

4.1. 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 as rules in the corresponding knowledge bases. Mechanical functions and behaviors of a product are usually realized through the interactions between pairs of critical faces. These interactions are specified 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. Other specifications are features' attributes or constraints. Each conceptual feature instance records its owning functions as well as the behaviors it participates in. Conceptual features associate to each other through their owning functions or behaviors. The conceptual design can be regarded as a skeleton of the product, which consists of abstracted critical faces, constraints, and other specifications (functions and behaviors).

4.2. Detail design feature model

In this stage, the complete product geometry and specifications are developed according to the skeleton specified during the conceptual design stage. Traditional form features are applied mainly in this domain. In the proposed unified feature modeling scheme, when constructing a detail design feature model, the specified conceptual design features are associated with the corresponding detail 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 detail design feature model, such as the relative position, orientation, motion, connection and fit relations. The detail design can be seen as a refinement and concretization process, where detail design feature model is enriched gradually

by the embodiment from critical faces into parts and 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 associations are registered in the *UnifiedFeatureModel* object.

4.3. Process planning feature model

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

The geometry entities of a process planning feature may correspond to one or more faces in the final or intermediate product geometry. The process planning features associate to other objects, such as machine, tool, setup and process planning rules, to represent a feasible process plan. Similarly, the associations between the detail design features and the corresponding process planning features are stored in the *UnifiedFeatureModel* object.

5. Change propagation

The inherent inter- and intra-stage relations analyzed in the previous section are represented as associations in the unified feature modeling scheme. From the perspective of implementation, there are two basic types of associations: constraint-based associations and sharing associations.

The constraint-based associations describe the geometric or non-geometric dependency relations between entities. For example, besides the traditional geometric or algebraic constraints, the dependency relations between the antecedent and consequent facts of a fired rule can also be represented as a constraint. The constraint-based associations are recorded using a Justification-based Truth Maintenance System (JTMS) [32] and handled by the numerical constraint solver or the rule-based expert system.

The sharing associations represent that two features refer to the same geometric or topological entities in the product master geometry. The sharing associations are handled by the unified cellular model. The sharing associations can be used to maintain the geometric consistency among lifecycle stages.

These associations are used to construct a dependency network for change propagation and information consistency control. Based on these associations, an algorithm for change propagation within and across stages is presented as follows.

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

(1) 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.
(2) Check the constraints between each v_i and its driving or driven variables one by one:
   – 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
       ○ If the value of the driven variable is fixed by the constraint, i.e. without alternative values, then the modification is rejected, run Abandon().
       ○ 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 related variable is a driving variable, then Abandon().
   – 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 check 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* */

(1) 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. The members of set_3 are checked (in the next two steps) one by one until set_3 becomes empty.
(2) The values of members of set_3 are temporarily changed in the database using the constraints recorded in the calling application.
(3) 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.

- If all modifications are accepted, the initial modification in the calling application is globally accepted and committed.
- 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* */

(1) All modifications made in the calling and called applications are revoked using backup values.

(2) In the database, the data of the called application, whose values are temporarily changed, are set back to their original values.

This algorithm aims at improving the efficiency of change propagation during the product modification process. Without recording dependency relations within a product model, the whole product model must be redeveloped when a modification occurs locally or in other related applications, because it is hard to identify the influence scope of the modification. With the proposed JTMS-based dependency network and change propagation algorithm, it is possible to narrow down the search scope faster and find out more accurately what other elements in the product model are affected by this modification.

6. Case study

In this section, two case studies are used to illustrate the advantages of the proposed dependency network and change propagation algorithm.

The first case is a cooling system design in an injection mold assembly (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 stage, cooling circuits may 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 stage 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 may invoke a chain of changes of the remaining cooling components. Without an efficient and effective integrity control mechanism, managing these relations and hence maintaining information consistency will be time-consuming and error-prone. This case study is implemented using Unigraphics WAVE technology. Interesting readers may refer to [27] for details of implementation.

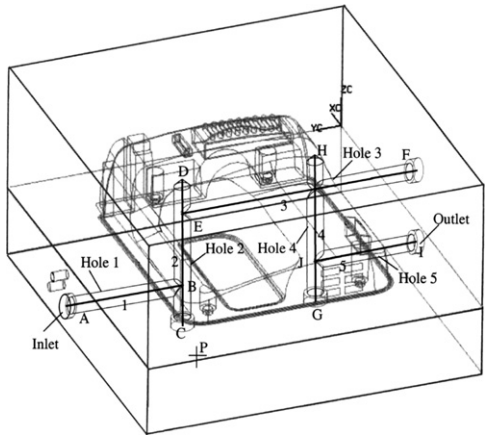


Fig. 4. A cooling system design in an injection mold assembly [27].

The second case study is an ejection system design in an injection mold assembly. This case study is implemented using ACIS [33]. MySQL [34] is used to develop the relational database, in which the public data of each application as well as the inter-stage associations are stored.

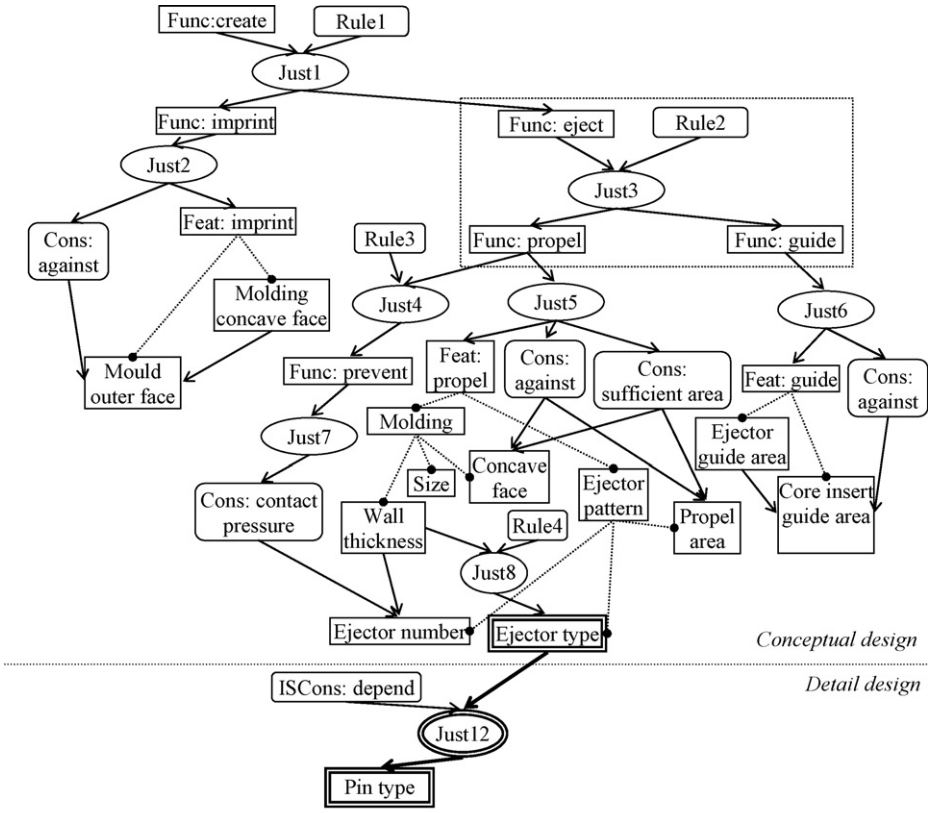
Along with the conceptual and detail design process, a JTMS dependency network is established (Fig. 5 is for conceptual design stage and Fig. 6 for detail design stage).

The inter- and intra-stage associations are established and stored explicitly in the product model. In Figs. 5 and 6, 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 Fig. 5, the portion surrounded by the dashed square represents a primitive reasoning process: based on the input function ''*eject* the molding'' and ''Rule2'' as antecedents, justification ''Just3'' generates the two sub-functions, ''*propel* the molding'' and ''*guide* the ejector'' as the consequents. The whole dependency network is established using these primitive reasoning blocks. The detail design of the ejection assembly is shown in Fig. 7.

In this case, the molding is of box-type (Fig. 8(a)). The original normal ejector pins can provide sufficient contact area between the pins and the molding (Fig. 8(b)). When the designer changes the wall thickness of the molding, the detail design application uses its dependency network to find out the affected features. The double-lined entities and justifications in Figs. 5 and 6 represent the change propagation chain. In this case, the wall thickness is reduced such that the algebraic constraint of ''sufficient contact area'' between the walls of the molding and the pin head faces is violated (Fig. 8(c)).

In general, several counteractions to increase the contact area between the molding and the ejector pins exist: changing the positions, diameters, or the type of pins (Fig. 9).

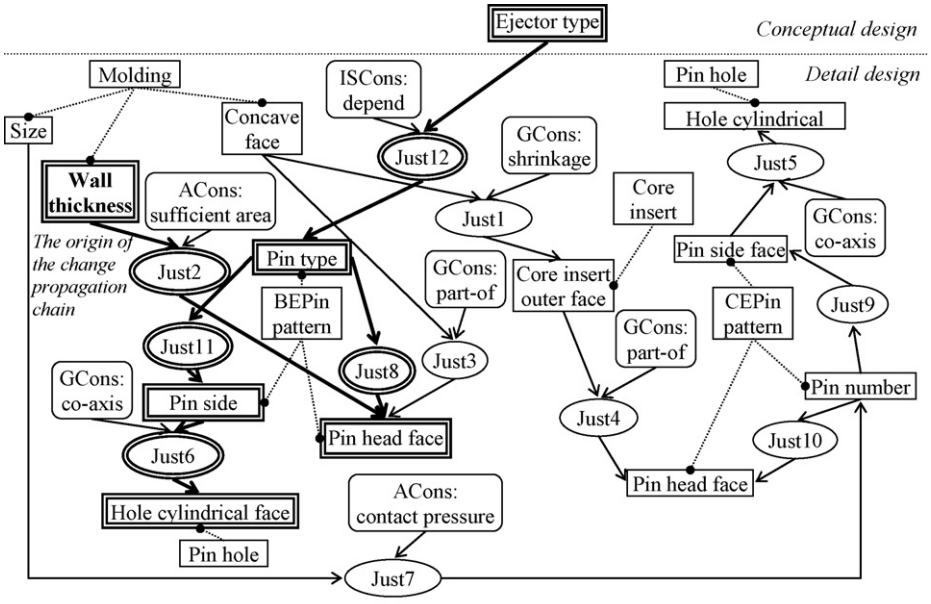
However, in this case, the first two choices are infeasible due to the pins' interference with the core insert. Changing the pin type is a feasible solution. Compared with the normal ejector pins, commonly used D-shape ejector pins are chosen here because they can provide bigger contact area (with the same pin diameter) to eject thin-wall molding. 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. The association is established during the detail design process. This modification is hence transferred to the conceptual design application for validation. 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 (Fig. 10). As for the detail design application,



Func – function; Feat – feature; Just – justification; Cons – constraint; ISCons – inter-stage con-

from an entity to its properties ⟶ from antecedents to justifications to consequents

Fig. 5. The dependency network for the conceptual design of an ejection system.



Just – justification; ACons – algebraic constraint; GCons – geometric constraint; ISCons – inter-stage constraint; BEPin – border ejector pin; CEPin – central ejector pin

from an entity to its properties ⟶ from antecedents to justifications to consequents

Fig. 6. The dependency network for the detail design of an ejection system.

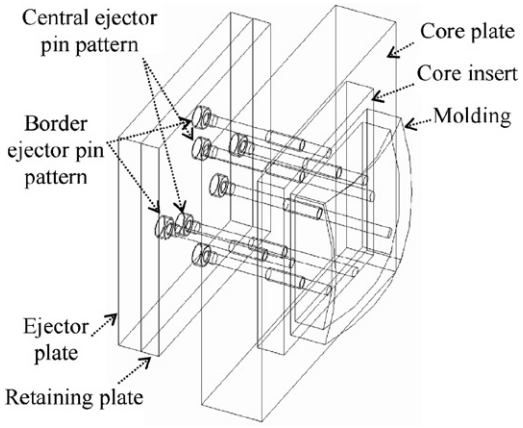


Fig. 7. The detail design assembly of an ejection system.

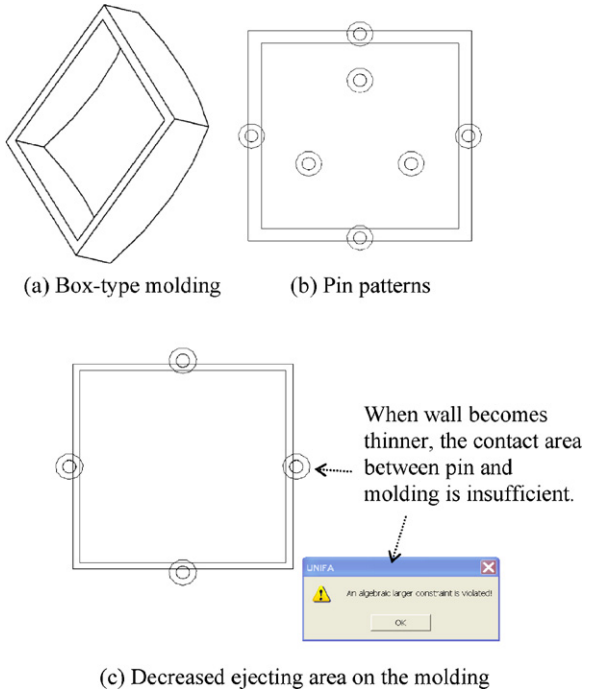


Fig. 8. Changing the wall thickness of the molding invalidates the ‘‘sufficient contact area’’ constraint.

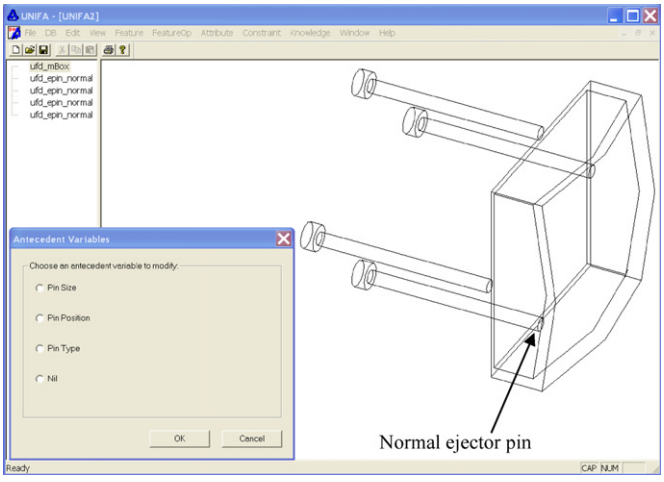


Fig. 9. Choices for re-satisfying the ‘‘sufficient contact area’’ constraint.

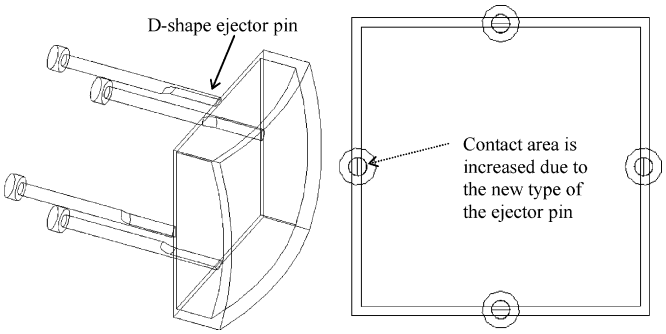


Fig. 10. The pin type is change to ‘‘D-shape’’ to increase the contact area.

changes of 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. Conclusion

Product lifecycle stages are inter-related and mutually constraining. To achieve an overall satisfied performance of a product, these stages must be managed as a coherent whole. Because of the unavoidable iterations among lifecycle stages, intra- and inter-stage relations must be identified, modeled and maintained carefully. This paper presents a change propagation algorithm in a unified feature modeling scheme. In this scheme, the whole product information model is represented as a dependency network. Different application feature models relate to each other through direct constraint-based associations as well as indirect common geometry-based sharing associations. Geometric and non-geometric associations are handled uniformly in the scheme. Modifying a single node (a variable) in this dependency network may have a sequence of influences. The proposed algorithm aims at maintaining the validity and consistency of the product models in the collaborative and concurrent engineering in a more efficient way.

References

[1] G. Thimm, G.A. Britton, 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 24 (3-4) (2004) 261–271.
[2] G. Thimm, G.A. Britton, 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 24 (3–4) (2004) 272–278.
[3] C.M. Eastman, Managing integrity in design information flows, Computer-Aided Design 28 (6/7) (1996) 551–565.
[4] G. Chen, Y.-S. Ma, G. Thimm, S.-H. Tang, Unified feature modeling scheme for the integration of CAD and CAx, Computer-Aided Design & Applications 1 (1-4) (2004) 595–602.
[5] G. Chen, Y.-S. Ma, X.-G. Ming, G. Thimm, S.G. Lee, L.-P. Khoo, S.-H. Tang, W.-F. Lu, A unified feature modeling scheme for multi-applications in PLM, in: M. Sobolewski, Q. Ghodous (Eds.), 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, ISPE, Dallas, USA, 25–29 July, (2005), pp. 343–348.

[6] G. Chen, Y.-S. Ma, G. Thimm, S.-H. Tang, Knowledge-base reasoning in a unified feature modeling scheme, Computer-Aided Design & Applications 2 (1–4) (2005) 173–182.
[7] T. De Martino, B. Falcidieno, S. Habinger, Design and engineering process integration through a multiple view intermediate modeler in a distributed object-oriented system environment, Computer-Aided Design 30 (6) (1998) 437–452.
[8] S. Subramani, B. Gurumoorthy, Maintaining associativity between form feature models, Computer-Aided Design 37 (13) (2005) 1319–1334.
[9] W.F. Bronsvoort, A. Noort, Multiple-view feature modeling for integral product development, Computer-Aided Design 36 (10) (2004) 929–946.
[10] M. Schulte, C. Weber, R. Stark, Functional features for design in mechanical engineering, Computers in Industry 23 (1–2) (1993) 15–24.
[11] M.R. Henderson, Representing functionality and design intent in product models, in: Proceedings on the Second ACM Symposium on Solid Modeling and Applications, Montreal, Quebec, Canada, (1993), pp. 387–396.
[12] J.-K. Gui, M. Mantyla, Functional understanding of assembly modeling, Computer-Aided Design 26 (6) (1994) 435–451.
[13] M. Ranta, M. Mantyla, Y. Umeda, T. Tomiyama, Integration of functional and feature-based product modelling—the IMS/GNOSIS experience, Computer-Aided Design 28 (5) (1996) 371–381.
[14] S.R. Gorti, A. Gupta, G.J. Kim, R.D. Sriram, A. Wong, An object-oriented representation for product and design processes, Computer-Aided Design 30 (7) (1998) 489–501.
[15] A. Chakrabarti, T.P. Bligh, A scheme for functional reasoning in conceptual design, Design Studies 22 (6) (2001) 493–517.
[16] S.F. Qin, R. Harrison, A.A. West, I.N. Jordanov, D.K. Wright, A framework of web-based conceptual design, Computers in Industry 50 (2) (2003) 153–164.
[17] N. Aleixos, P. Company, M. Contero, Integrated modeling with top-down approach in subsidiary industries, Computers in Industry 53 (1) (2004) 97–116.
[18] Y.-H. Han, K. Lee, A case-based framework for reuse of previous design concepts in conceptual synthesis of mechanisms, Computers in Industry 57 (4) (2006) 305–318.
[19] W. Van Holland, W.F. Bronsvoort, Assembly features in modeling and planning, Robotics and Computer Integrated Manufacturing 16 (4) (2000) 277–294.
[20] R.B. Gottipolu, K. Ghosh, A simplified and efficient representation for evaluation and selection of assembly sequences, Computers in Industry 50 (3) (2003) 251–264.
[21] K.-Y. Kim, Y. Wang, O.S. Muogboh, B.O. Nnaji, Design formalism for collaborative assembly design, Computer-Aided Design 36 (9) (2004) 849–871.
[22] B. Khoshnevis, D.N. Sormaz, J.Y. Park, An integrated process planning system using feature reasoning and space search-based optimization, IIE Transactions 31 (7) (1999) 597–616.
[23] R. Stage, C. Roberts, M. Henderson, Generating resource based flexible form manufacturing features through objective driven clustering, Computer-Aided Design 31 (2) (1999) 119–130.
[24] C.-C.P. Chu, R. Gadh, Feature-based approach for set-up minimization of process design from product design, Computer-Aided Design 28 (5) (1996) 321–332.
[25] A.S. Kumar, A.Y.C. Nee, S. Prombanpong, Expert fixture-design system for an automated manufacturing environment, Computer-Aided Design 24 (6) (1992) 316–326.
[26] S.R. Subrahmanyam, Fixturing features selection in feature-based systems, Computers in Industry 48 (2) (2002) 99–108.
[27] Y.-S. Ma, T. Tong, Associative feature modeling for concurrent engineering integration, Computers in Industry 51 (1) (2003) 51–71.
[28] Y.-S. Ma, T. Tong, An object-oriented design tool for associative cooling channels in plastic-injection moulds, International Journal of Advanced Manufacturing Technology 23 (1–2) (2004) 79–86.
[29] A. Kusiak, J. Wang, Dependency analysis in constraint negotiation, IEEE Transactions on Systems, Man, and Cybernetics 25 (9) (1995) 1301–1313.
[30] H. Park, M.R. Cutkosky, Framework for modeling dependencies in collaborative engineering processes, Research in Engineering Design 11 (2) (1999) 84–102.
[31] G. Booch, J. Rumbaugh, I. Jacobson, The Unified Modeling Language User Guide, Addison Wesley, 1999.
[32] K.D. Forbus, J. de Kleer, Building Problem Solvers, MIT Press, 1993.
[33] Spatial Corp., ACIS 3D Geometric Modeler, Version 7.0, http://www.spatial.com, 2001.
[34] MySQL AB, MySQL Reference Manual for Version 5.0.1-alpha, http://dev.mysql.com/doc/mysql.

Yongsheng Ma has been an associate professor at the school of Mechanical and Aerospace Engineering, Nanyang Technological University (NTU), Singapore, since 2000. His main research areas include product lifecycle management, feature-based product and process modeling. Dr. Ma received his BEng from Tsing Hua University, Beijing (1986). He obtained both his MSc and PhD degrees from Manchester University, UK in 1990 and 1994, respectively. Before joining NTU, he was a group manager with the Singapore Institute of Manufacturing Technology.

Gang Chen is currently a research project officer and a PhD candidate in School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore. He graduated from Henan University of Science & Technology and received BEng degree in 1993. From 1993 to 2001, he worked as a vehicle test engineer in Auto Testing Research Institute of China Automotive Technology & Research Center. He entered Nanyang Technological University in January 2002.

Georg Thimm joined the Dalle Molle Institute for Perceptual Artificial Intelligence in Switzerland after receiving in 1992 a diploma in computer science from the University of Karlsruhe. There, he performed research on higher order neural networks in preparation of a doctor degree in technical sciences, which he obtained in 1997 from the Swiss Federal Institute of Technology in Lausanne. He joined the Nanyang Technological University (Singapore) in 1999 as research Fellow, was converted to assistant professor in 2000 and is Director of the Advanced Design and Modeling laboratory. His research interests are in the application of artificial intelligence and graph theory, including process planning, product life-cycle management, and crystallography. He is member of several editorial boards of internationally recognized journals.