ELSEVIER

Computers in Industry 51 (2003) 51–71

COMPUTERS IN INDUSTRY

www.elsevier.com/locate/compind

Associative feature modeling for concurrent engineering integration

Y.-S. Ma[*], T. Tong

School of Mechanical and Production Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798, Singapore

Received 22 November 2001; accepted 1 February 2003

Abstract

In typical product development processes, like plastic injection mould design, design information flow is not well supported by the current available IT systems. At different stages of a product life cycle, from documentation of requirement specifications, to conceptual design, detailed structure design and production, engineering knowledge is striped off except the bare minimum geometrical and control data, such as computer-aided design (CAD) solid models and cutting tool paths. Associative relations among engineering features are normally ignored; hence data consistency and design changes are difficult to be managed. In this paper, interfacing knowledge oriented tools and CAD application is identified as a technical gap for intelligent product development, and the concept of associative feature is introduced. For high-level reasoning and the execution of decisions, to define associative features in the form of self-contained and well-defined design objects is essential. As a case study, cooling channels in the design of plastic injection moulds with a CAD tool are modeled as an illustrating associative feature type. Potential integration among different applications based on such associative features is also explored.

Keywords: Associative features; Knowledge engineering; Design automation; Injection mold design; CADCAM

1. Introduction

In concurrent engineering [1], all the engineering processes are supposed to be supported with integrated computer-aided tools, and based on a consistent set of data with different application views. Such applications include conceptual design, structural design, detailed design, design analysis for certain specific engineering aspects (DFX), computer-aided process planning (CAPP), and computer-aided manufacturing (CAM) tool path generation, etc. Unfortunately, this proposed scenario has not been realized due to the interoperability limitations of different software packages. So far almost all the computer-aided design (CAD) system vendors have only implemented data exchange functions for un-parameterized two-dimensional (2D) drawings or 3D solid models. Such rigid geometry models are then used for other applications. International effort has been made to develop a standard, STEP, for engineering data representation and communication [2,3]. Although, it has contributed in a great way for product modeling, however, implementing this standard in real applications still requires a lot of information modeling and

[*] Corresponding author. Tel.: +65-6790-5913;
fax: +65-6791-1859.
E-mail address: mysma@ntu.edu.sg (Y.-S. Ma).

development, especially for the interrelated objects that reflects different perspectives and abstraction levels within the design domain [4].

It has been acknowledged that in concurrent engineering [5], interoperability should cover the relationship between a knowledge-based engineering (KBE) system and a CAD platform. However, to transfer information from CAD to KBE systems is very difficult because KBE systems rely heavily on the design intent to perform activities, such as cost estimation or DFX analyses. The intelligence added to CAD geometry is either stripped off by the translation software or unrecognizable by KBE system. In addition, many CAD systems are unable to completely and unambiguously capture design intent. On the other hand, transfer KBE intelligence to CAD systems is equally challenging because there is no mechanism to enable such information flow.

It has been highlighted in concurrent engineering that product and process models need to be addressed at three different levels of representations, knowledge, information, and data levels [1]. Geometrical entities are complex in nature and are usually integrated with KBE systems via another layer of object-oriented software in order to achieve effective reasoning and execution [1,6–8]. This is due to the fact that most knowledge-oriented systems use first-order logics as the foundation, and each predicate has to be self-contained, and well defined. The format of predicates with objects can be as:

Bigger-than (Object-A, Object-B); or
Bigger-than (x, bigger-of (Object-A, Object-B)).

One way to bridge these gaps is to build high-level design expertise and rules in a knowledge-oriented system [9,10] while low-level design intent into a CAD system in the form of information-rich objects, which can be referred to as features in general although there have been many different definitions in the past literature [11–14]. As shown above, with certain naming conventions, such objects can then be mapped as arguments of predicate calculus, and manipulated with artificial intelligence (AI) languages, such as Prolog. Knowledge-driven queries and operations on these objects become feasible. Once a complete product model is fully defined with self-contained objects, concurrent engineering throughout the product life cycle can be supported via aspect models and meta-models [7,12,15].

In this paper, interfacing knowledge-oriented tools and CAD applications is identified as a technical gap for intelligent product development, and a new concept, named associative feature, is introduced. The authors intend to expand the existing feature types to include a flexible group that has imposed great difficulties in traditional feature based design. For high-level reasoning and the execution of decisions, to define associative features in the form of self-contained and well-defined design objects are essential. As a case study, cooling channels in the design of plastic injection moulds with a CAD tool are modeled in the form of associative feature type. Potential integration among different applications based on such associative features is also explored.

This paper consists of seven sections including this introduction. In Section 2, the feature technology is briefly reviewed. Sections 3 gives a definition for the new feature type, namely associative feature. Then in Section 4, the design of cooling channel representation, after analyzing the detailed requirements, is presented. Section 5, the proposed solution and functionality are introduced. In Section 6, potential integration with other applications is discussed. Finally, summary discussion and conclusions are given in Section 7.

2. Feature technology review

There have been many kinds of definitions for features used in CAD/CAM arena. They are initially modeled based on machining features that can be used to integrate CAPP and CAM packages [11]. After many years of development on feature-based design or manufacturing, most of the implemented features are still CAD application oriented and related to machining processes (e.g. holes, slots, pockets, etc.) or design geometry (e.g. drafting angle). Their definitions are based on predefined parametric templates, which do not have the flexibility to be extended or reconfigured as required by some design processes, like mold-cooling channel design. Otto [16] recognized three groups of information consistency breakdowns to be addressed by feature technology, static, dynamic and hybrid breakdowns. Effort has been tried to overcome these issues, such as CAD and CAE feature information sharing [8]. For example, when plastic parts are to be analyzed with CAE applications, walls, ribs, middle planes are concerned; hence, they are defined as features. More recently, feature concept has been very much expanded to any meaningful grouping schema related to geometrical entities. In fact, feature definition is very much dependent on the application purposes. Otto [16] described the role of features is to relate (product or part) geometry to an engineering discipline in order to represent certain meaning. Feature modeling represents a special application of information modeling. In this paper, the authors intend to differentiate a new group of features that has not been addressed in research papers.

3. The concept of associative features

Use mold design as an example. Certain geometrical entities are grouped with specific characteristics. These entity groups can be identified as core and cavity inserts [17], sub-inserts and electrode [18], cooling channels [19,20], and machining set-ups [21]. Let us analyze such geometrical entities briefly. Core and cavity inserts enclose a cavity volume in a plastic injection mold that forms the shape of a molding part. Usually, the molding part is designed first by product designers. Core and cavity inserts shares molding part surface geometry that is separated by the parting line [22]. Hence, in order to support the consistent design throughout the concurrent engineering life cycle, core/cavity surfaces should be associated with the molding part surfaces, so that, if the part geometry is modified at later stage, the mold design can be updated accordingly, and changes can be minimized. Sub-inserts are used when undercuts exist on the molding part geometry and side-cores or side-cavities have to be used. Again, a sub-insert's impression face is associated with the molding part geometry. Such association has not been modeled concisely as a feature or an object [18]. The similar relations exist between a mold electrode's geometry and the molding part because it is just the inversed definition to sub-inserts. Cooling channels are also supposed to be inter-connected to form cooling circuits. When mold designer carries out conceptual design, cooling circuits are considered. However, when potential geometrical collision between cooling channels and other features being checked, 3D cooling channel geometry must be considered. For cooling effect analysis, CAE tools also require cooling circuit mesh representation [19]. This paper uses cooling channels as the illustration case; its associative nature will be introduced in detail in the following sections.

It can be observed that such design elements' topological configurations cannot be predefined, and yet very much commonality exists among different instances. The intricacy of geometrical relations in the term of mutual dependency is essential. They are named as associative features.

Associative features are very difficult to be represented with the traditional feature concept because as such they cannot be wrapped up as predefined two-manifold entities [16]. Hence, whenever such features are involved, there exist difficulties to fully integrate product design models with high-level knowledge representation models [12,21].

To represent associative features consistently, object-oriented technology offers excellent solution. In this work, a feature is defined as an object that relates geometrical entities, and supports all applications within its scope of purpose. The previous section has highlighted that features have to be flexible, self-contained, and consistent to integrate different applications. Here, "flexible" means the capability of the feature data structure that can be created, edited, and deleted by the end user dynamically through certain composing methods. The data structure should also provide interfaces to support different applications. "Self-contained" means features can keep its validation and integrity before and after any interaction with any integrated applications. In other words, if a feature can be represented with a self-contained object, it is believed to be well-defined [23]. It can be further appreciated that with a well-defined object model, features can be expressed as declarative knowledge and hence enables higher level knowledge-oriented processing [16].

To address the requirements for concurrent engineering and product life cycle support, the proposed model for associative features should have the following key characteristics:

- Built-in associative links to its related geometric entities.
- Self-validation for the consistency of its entities, attributes, constraints, etc.

- Methods available for constructing, storing, indexing, editing and destroying its instances.
- Methods that can be expanded to interface with query and execution mechanisms for high-level knowledge processes.
- Methods to interface with other engineering application tools.

Based on the above concept, an associative tool for cooling system design is developed as a module of MoldWizard^{TM}, which is a special process-based software solution for plastic injection mold design.

4. Design of associative cooling channels

4.1. Current practice

Due to the short product development cycles, plastic injection mold designers are required to compress their design time and to accommodate more late changes. The design of cooling system affects not only the quality of mold assemblies but also the efficiency in manufacturing. Many factors must be considered in designing a cooling system, they include molding part thickness, cooling channel number, locations and sizes. Li [24] proposed a design synthesis algorithm by decomposing a complex molding part shape into shape features and merging their individual cooling channels into acceptable conceptual cooling circuits. It has been indicated by Singh [25] that design variables, such as locations, types of cooling channels and 3D layout of circuits, are usually modified frequently for addressing late part design changes as well as mold design optimization. The modification process is laborious and error-prone because designers have to repetitively edit and update CAD models.

Computer-aided tools for injection molding has emerged since early 1970s [6], most of which are focused on optimization algorithms. This can be shown in the works about flow computation and analysis routines [26]. An expert system for designing mold-cooling systems was introduced by Kwon and Weeks [27]. The system consists of four modules, top-level layout design, analysis, evaluation and decision-making. A decision-making module through a design cycle evaluates the redesign of cooling channels based on the rules stored in a knowledge base. However, this system was not integrated with a CAD system; initial design parameters are input by the user via a command line interface. Wang and his co-workers [19,28] had developed a computer-aided mold design system. They suggested a design strategy with three-stages, initial design with one-dimensional (1D) approximation, two-dimensional design with optimization, and three-dimensional design with cooling effect analysis. They had developed Cornell Cooling System Design Program (CCSDP), which uses 3D boundary element method (BEM) to analyze 3D heat-transfer. In their work, they recognized the fact that the parameters in designing mold-cooling system are numerous. They listed eight design modification options if the cooling system is not satisfactory. They are all related to cooling channels geometry changes. That is why associative cooling design tools are urgently required. They should be able to free mold designers from tedious geometrical updating and to keep design models consistent so that the total mold design cycle time can be shortened.

Mok et al. [29] had considered cooling systems with automated retrieval of certain circuit patterns, such as straight or U types. Since they used fixed patterns, the system does not provide the flexibility to compose complex cooling circuits. More focused areas for the creation of sub-systems, such as core/cavity parting [17,30,31]; runners [32,33], gate locations [26] and cooling systems [27] are also studied. However, the association among geometrical entities is not discussed.

More recently, object-oriented (OO) software development technology has been applied in the development of mold design tools [6,34,35]. Object definitions help in great deal to sort out complicated relationships in mold design, especially among part-independent parts and features. However, keeping the relationships among geometrical entities and making them easily customizable is still not a trivial task.

The creation of cooling-holes/channels entails several time-consuming and manual tasks, such as creating holes, maintaining the connections among them to form circuits. If these tasks can be partially automated, the design time can be reduced considerably and hence the productivity enhanced. In industry practice, it is common to use at least four major cooling circuits in a mold assembly. They are located on, namely, the cavity insert, the core insert, and the A- and B-plates [22]. In most of the CAD design tools, a drilling hole is modeled as a HOLE feature, which is a negative cylindrical solid with/without a cone end. Other extended features include counter-bored hole, thread hole, etc.

However, because solid features are rigidly associated with the housing solid, it was found from very experienced mold designers that, solid cylinders instead of features are commonly used to represent cooling channels. In the case of blind channels, the cylinders are chamfered at the blind end to make them appear as drilled-blind holes. When the design is finalized, all channels are united to form a cooling circuit. These circuits are not converted into holes (or cuts) until the design has been finalized and ready for down stream tasks, such as thermal analysis or CAM cutting tool path generation.

More detailed reasons for not using HOLE features directly to represent cooling channels are in four folds:

- It is useful for visual inspection to plot a drawing, which has highlighted cooling circuits (without detailed features of the cavity and core blocks, and mold plates). It is not possible to plot cooling circuits only if they are made up of HOLE features created on solids without showing the details of these solids.
- Repositioning and modifying cylinders in a CAD environment require much less steps than HOLE features.
- When HOLE features is used, when plotting cooling circuit drawings, they cannot be assigned with different colors from their belonged solids. This gives difficulties in checking cooling circuits against other components or features on a plotted drawing; even through the drawing can be plotted in color.
- It is not easy if not impossible to automatically check collisions with other mold components, e.g. ejection pins and cooling channels, if they are represented with HOLE features.

On the other hand, using native CAD modeling functions to create cooling channels in the form of solid cylinders has several shortcomings. Firstly, many steps are required for a simple channel:

- Create a cylinder,
- chamfer the blind end if it represents a blind channel,
- run through equal series of dialogue boxes to position, copy and re-orient the cylinders, if needed.

Secondly, no intelligence is built-in in the cooling channel representation. For example, it is good to have basic reasoning capabilities like:

- Identifying cooling channel cylinders (not any cylinder in the model). This is particularly important when the cooling channel information is to be used for heat-transfer analysis and collision check.
- Providing orientation information for plugs to be inserted into cooling channels with user-friendly drag and place manner.

Thirdly, all the geometrical elements created are separated and individual without organization and association. Therefore, if there is any change at the late stage of design, a lot of modifications are required. They are error prone. In most cases, late changes are inevitable and frequent, such that the mold designer would be frustrated with the tedious modifications and corrections. Some errors can cause mold delay or poor assembly quality.

4.2. Expectation for associative cooling channels

To overcome these shortcomings, there is a need to have a special software tool for designing cooling channels in a mold assembly. This work is targeted to provide substantial automation for cooling circuit generation. Besides, it needs to provide associative links among cooling-holes and their relevant faces. This is the most important objective of this work except user-friendly process automation. It can be comprehended that if all the cooling channels are created in an associative approach, then declarative terms in knowledge rules, such as distances from cavity features, attached component orientations, penetrating faces and drilling directions, and the connectivity within a circuit can be embedded with the CAD models. Hence, the information stored becomes explicit and persistent. The cooling channels are to be updated if penetrating faces or hole types need to be modified at later design stage. Comparing to the approaches in [27,29], this approach has a great advantage that the mold designer can accommodate modifications easily throughout the product life cycle. It can also be said that, design rules stored in a separate knowledge base, can be related to cooling feature objects embedded associatively in the CAD geometrical database.

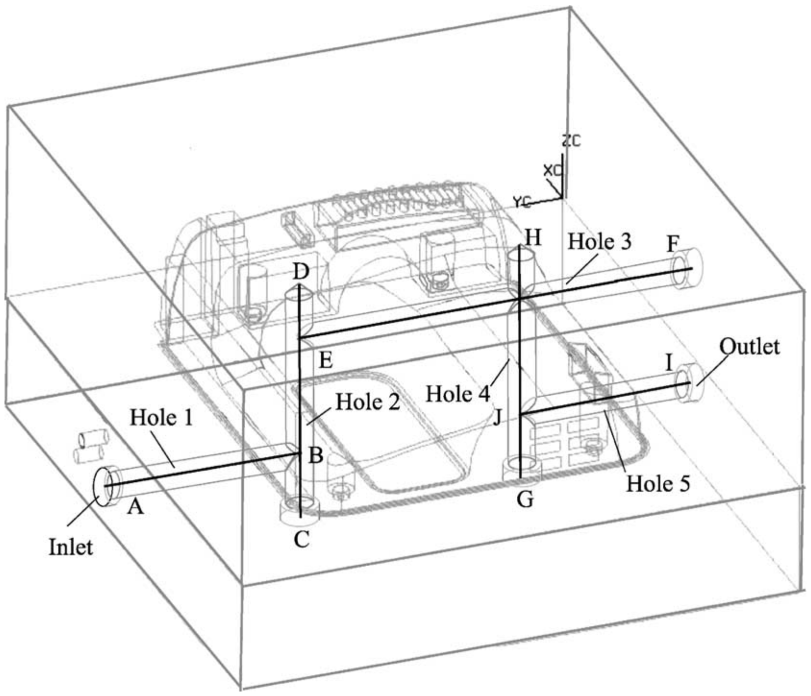


Fig. 1. Example of a cooling circuit.

4.3. *Definitions for associative cooling channels*

In this work, the cooling system of a mold consists of several cooling circuits, and a cooling circuit consists of a few cooling-holes. An example of cooling circuit is shown in Fig. 1. A cooling circuit is defined as a type of objects that represent all the interconnected cooling-holes (Hole 1 to Hole 5 in Fig. 1) with different orientations between a cooling inlet and its corresponding outlet. Fig. 2 shows the semantic structure of cooling system objects. Here, ‘‘hole’’ is used to describe the geometrical shape of a cooling channel, however, its representation is not the traditional HOLE features. These holes are drilled from faces of different mold plates or inserts. The face used to drill a cooling-hole is named as penetrating face. Naturally, a cooling-hole has one penetrating face and hole-drilling direction is always leaving from the penetrating face and pointing to the other end. Usually, cooling-holes are perpendicular to the penetrating face.

To represent cooling-holes, cooling guiding lines have to be introduced. Some terms used are shown in Fig. 3. A cooling-hole has a guiding line, which is a straight-line segment starting from a cooling-hole penetrating center point to the other end center point. In a design session with this cooling module, cooling-holes are initially represented with their guiding lines with attached attributes to specify the ‘‘hole’’ parameters. For example, in Fig. 1, AB is the cool guiding line for Hole 1, and CD is for Hole 2. Note that guiding lines also indicate hole-drilling directions. A cooling guiding line can be used to store certain attributes, such as the cooling-hole types, and diameters.

These guiding lines are used to generate cooling-holes; this point will be introduced in later sections. The cooling guiding lines of all the holes within a cooling circuit are grouped as a guiding path. For example, given in Fig. 1, the only guiding path for the circuit includes five guiding lines, i.e. AB, CD, EF, GH, and IJ. In this work, a guiding path essentially represents each cooling circuit (see Fig. 2). The information stored in a cooling guiding path is complete; cooling solids can be regenerated again and again. For a circuit, the guiding path is used to keep the

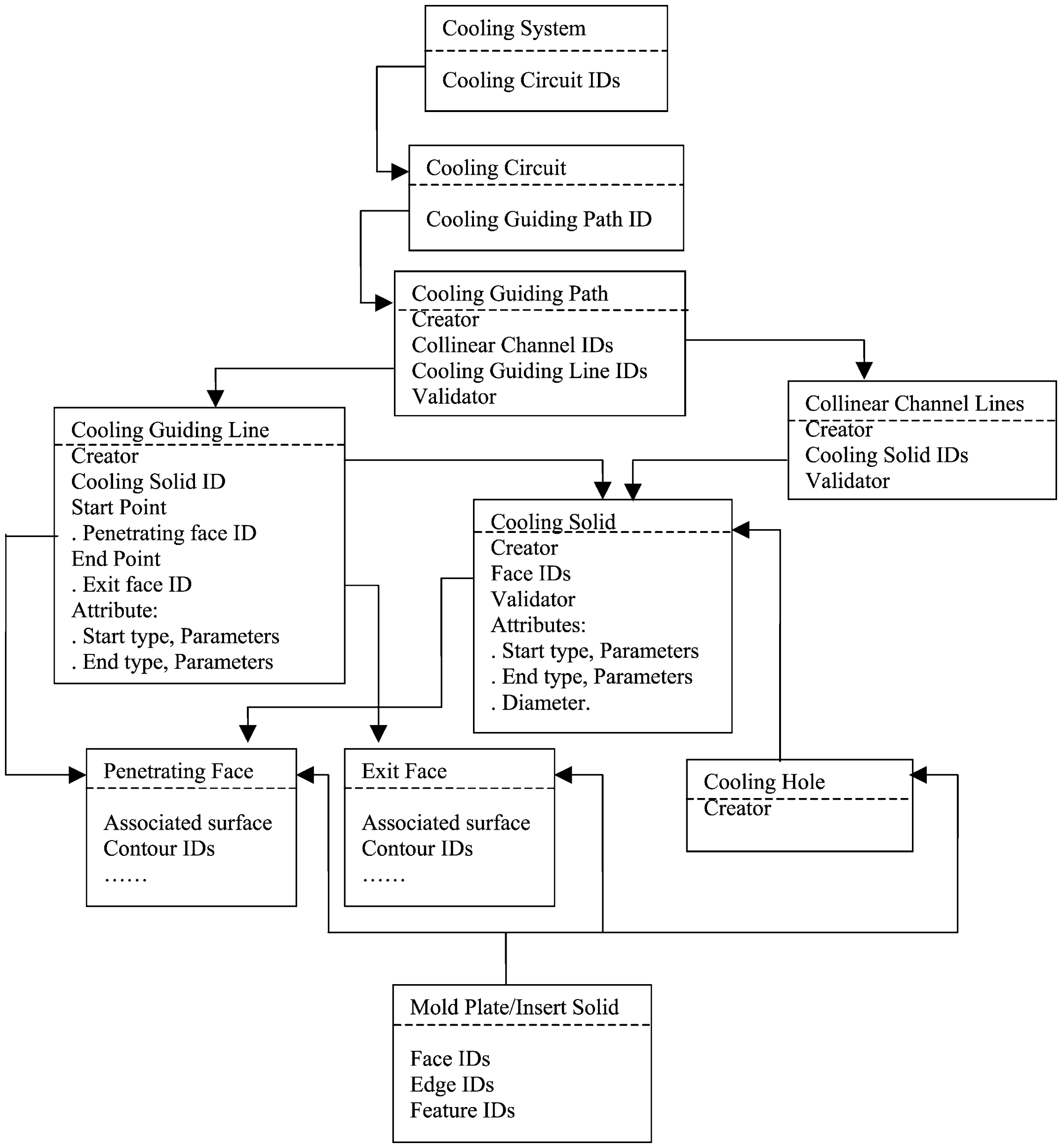


Fig. 2. Semantic structure of the cooling channel components.

connectivity among its guiding lines. To validate and verify this condition, a "validator" method is defined in the guiding path object.

To illustrate the volume in the design space, or to check physical collision among mold assembly elements, cylindrical solids (with cone ends in cases of blind holes) are generated automatically by a method based on individual guiding lines at any time when required. They are referred to as cooling solids in the following sections. These cooling solids can also be deleted to simplify the display. If, after confirming the cooling system design, geometrical HOLES are still needed for CAM application and component structure detailing, they can be achieved by subtracting cooling solids from their corresponding plate/insert bodies.

At the both ends of each cooling solid, there can be four types of configurations to be selected (see Fig. 4): (1) drill-through; (2) counter-bored; (3) blind without

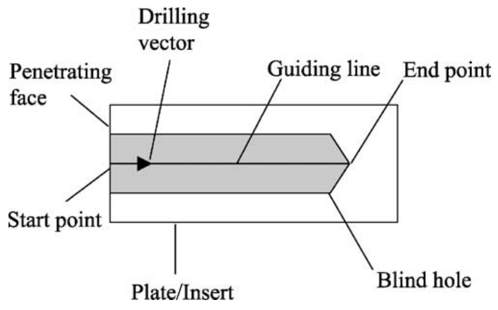


Fig. 3. Terms used for a cooling-hole.

extension and (4) blind with extension. These geometrical features can be represented with attributes attached to guiding lines. Such related attributes include types of the end, cooling-hole diameter, the depth and the diameter of the counter-bored hole if applicable. They are used for cooling-hole modification and cooling solid re-creation. In fact, a cooling guiding line contains all the information about each individual cooling-hole with attributes.

From Fig. 2, it can be seen that a cooling system contains all the geometrical and non-geometrical information and has much richer attributes than a group of holes or negative volume cylinders. In fact, it can be viewed as a package of knowledge combination in application domain. The contents and representation of a circuit object change according to the context and user's choices while only the essential persistent information is kept into the CAD database. For example, a circuit can be represented graphically as a set of inter-connected guiding lines, or just a set of cylindrical solids. Further elaboration can be seen in the functionality section.

In practice, very often, there are some cooling channels across multiple blocks (see Fig. 5). It means a cooling channel consists of several connected

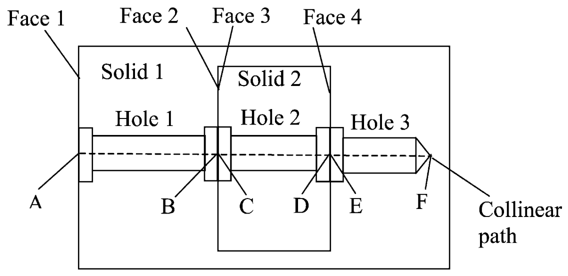


Fig. 5. A typical collinear cooling channel.

collinear cooling-holes (Hole 1, Hole 2 and Hole 3). To simplify the design, a special object type is created for this type of channels, named as "Collinear channel". From Fig. 2, it can be seen that a cooling circuit may contain several such collinear cooling channels other than simple cooling-holes. Each collinear channel is represented by a collinear path, which groups the entire collinear channel guiding lines. Each of its guiding lines still represents a cooling-hole in turn. In the case shown in Fig. 5, AB, CD and EF form a collinear path. Here, AB represents through Hole 1 with both counter-bored ends, CD represents through Hole 2, and EF represents the blind Hole 3. As discussed earlier, since, a cooling path can represent a cooling circuit, therefore, it can contain several collinear path other than simple cooling lines. Obviously, for collinear channel lines, their element guiding lines must be connected head to tail continuously.

For explaining how the cooling channels are linked to their corresponding penetrating faces, and then in turn, to the mold plates or inserts, the detailed internal data structure and relationships used in this work is given later.

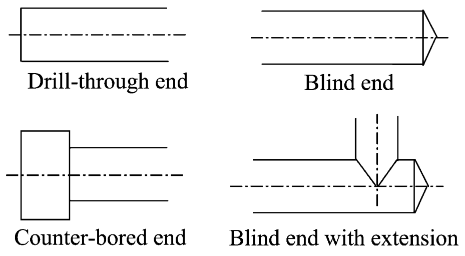


Fig. 4. Types of cooling cylinder ends.

5. Solution and functionality

5.1. Embedded links and parameters

In the implementation, the start and end points of a guiding line are made associated with the penetrating and exiting faces except the end point for a blind hole. To achieve this, "smart points" defined in Unigraphics^{TM} (UG) are used [36]. A "smart point" is "a point on surface" associated with the face at the kernel database level. It keeps the persistent link. If the face changes its position, the point will be derived dynamically and updated accordingly. In other words, the point follows the face changes. Hence, "smart" is referred to describe the associative nature of an entity. Since guiding lines are created based on such smart end points, then the corresponding guiding lines are also called as smart lines. For each of them, it is connected to one (in the cases of blind holes) or two smart points.

A cooling solid are generated along its smart guiding line with CAD application programming interface (API) functions. Initially, a cylindrical TUBE is generated by sweeping a circular section profile along a guiding line [36]; then end features are added. For example, to represent a blind hole, a cone end is added. These cylindrical solids are then united as the representation of cooling circuits.

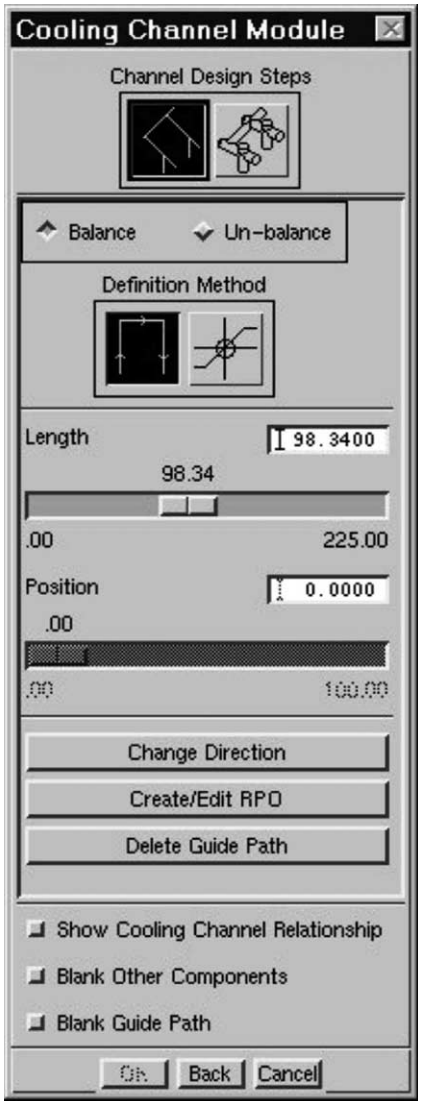
5.2. Functional design and algorithms

In order to serve the design requirements, this cooling channel module provides the following functions:

- Creation of a cooling circuit with a smart guiding path.
- Adding/Removing guiding lines to/from a guiding path.
- Modification/Reposition (RPO) of a guide path.
- Deleting of a cooling circuit and its guide path.
- Creation of cooling solids.
- Modification of the cooling solids.
- Deleting of a cooling solid.
- Create a reference set to which cooling-hole features on plate/insert are assigned.
- Deal with balanced and unbalanced designs for multi-impression mold.
- Some important functions of these are briefly described individually below.

5.2.1. Creation of a cooling circuit with a smart guide path

To create the first guiding line of a guide path, the cooling channel module is initiated first and the main UI is displayed (see Fig. 6). The user needs to select a face on an intended solid as the inlet penetrating (planar) face of the circuit (Fig. 7). A plane equation can be extracted from the selected planar face. Using



Length 98.3400
98.34
.00 225.00
Position 0.0000
.00
.00 100.00

Fig. 6. The UI for creating guiding lines.

the user's selection point returned, and the planar face normal vector, a drilling axis (a point and a vector) is formed. The drilling point becomes the initial start point for the guiding path. A smart point is created on the face. The default drilling direction (the direction to generate the first cooling guiding line) is set to the reverse direction of the face normal. An indicating arrow is displayed on the graphic window. The user can modify the direction, if he wants, with "Changing Direction" option from the UI (Fig. 6). Then, the user can dynamically drag a cooling line or input a value of

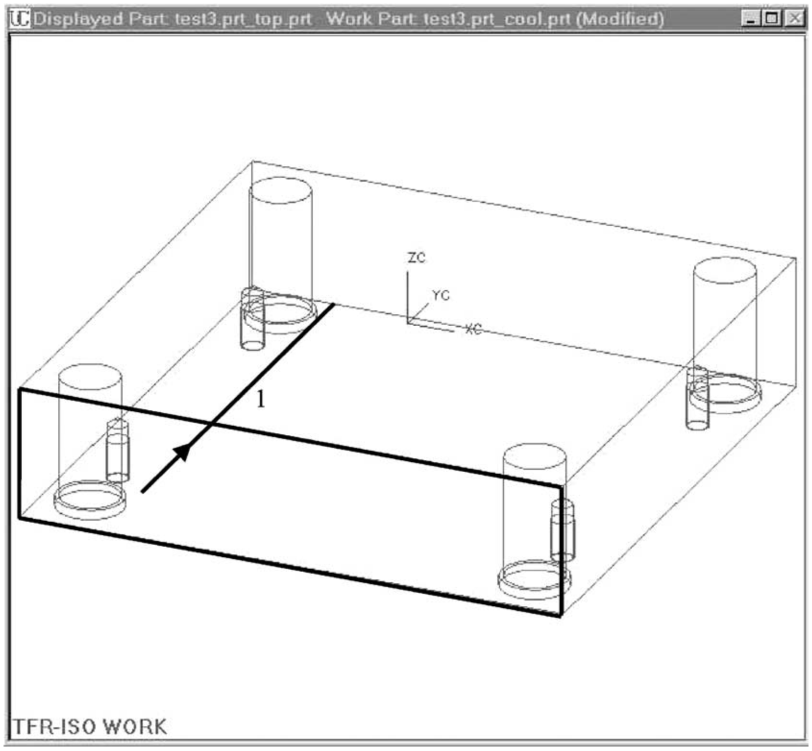

Displayed Part: test3.prt_top.prt Work Part: test3.prt_cool.prt (Modified)
ZC
YC
XC
1
TFR-ISO WORK

Fig. 7. Creation of the first cooling guiding line.

the length for the cooling guiding line. The user can also choose another face to indicate the ending face of a through "hole". In the latter case, another smart point is created at the end point of the guiding line. After creating the first guiding line, a sequence number "1" is displayed near it (Fig. 7). Note that the guiding line is a smart line.

To create the next guiding line (see Fig. 8), a drilling vector is still required. The user can indicate it by selecting a penetrating face at point F. Then, the next guiding line direction is set to be in the reverse normal direction of the selected face. The vector's start point C is found on the previous guiding line AB and is the nearest point to the user's indicated point F. This is an embedded rule implemented in this work. By default, to make the vector-definition user-friendly, many predefined "rules" are applied to assist guiding line creation. In the same example, when defining the next guiding line CD from the previous one AB, since there is no intersection between the new guiding line CD and the penetrating face, the guiding line is then extended automatically along the reverse direction, and made to start from the drilling point E on the penetrating face (Fig. 8). A smart point is created at E on this face to make the guiding line associated with the face. Similarly, another smart point D is created at the end point of the guiding line if it ends at a face (through hole only). Again, a sequence number is then displayed near the guiding line. In the similar manner, other guiding lines can be defined. Upon confirming all the guiding lines of the intended guiding path, the continuity within the path is checked by a method named as "validator" (see Fig. 2).

5.2.2. *Modifying and repositioning*

After a cooling guiding line is selected, its properties, including its length, can be edited with the UI as shown in Fig. 6. They can be changed and updated. In fact, when a guiding line is selected, its guiding path is also identified because all the guiding lines in a guiding path are associated with continuity constraints. If the inlet point position of the guide path is moved, the whole path follows accordingly. Surely the user can delete a guide path by selecting the

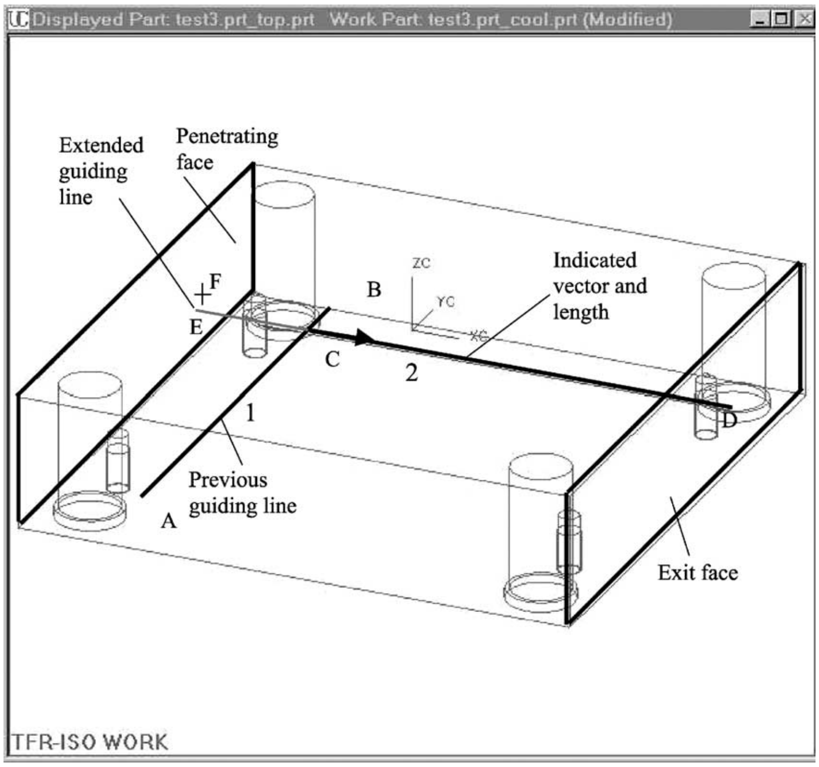

Displayed Part: test3.prt_top.prt Work Part: test3.prt_cool.prt (Modified)
Extended guiding line
Penetrating face
F
E
B
ZC
YC
XC
Indicated vector and length
C
2
D
1
Previous guiding line
A
Exit face
TFR-ISO WORK

Fig. 8. Creating another guiding line following a previous one.

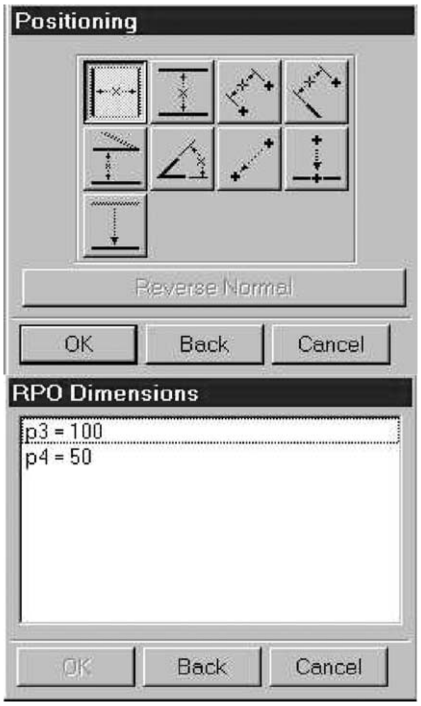
relevant option from the editing UI. Fig. 9 shows more editing options under Reposition (RPO) sub-menu. Fig. 10 is a screen snap.

5.2.3. Creating cooling solids

As described in the Section 4.2, cooling solids are created only when the user needs them. After a guide path is selected, the cooling solids can be created based on the attributes of its individual guiding lines. Cooling solid variations are defined as the start and the end features of their associated base solids. The UI for this purpose is shown in Fig. 11. Initially, the UI settings, such as start type, end type, hole diameter, are assigned with the default types and diameters preset in the UI configuration file. They are then updated based on the user's input, i.e. types and parameters (attributes) of the selected cooling guiding lines.

The configuration file also contains other initial values of the default ''hole'' types, such as counter-bore dimensions. Based on these default values, the corresponding UI text boxes can be initially set. The values in this configuration file are always overwritten with the user's choices of preferred values when he accepts the cooling solid creation UI dialog box. In this way, the UI settings become very user friendly with some ''learning'' capability. On the other hand, the entries to different fields of the dialog box are also verified against preset conditions.

Once these attributes are confirmed, cooling solids can be generated automatically by clicking ''Show Cooling Channel Relationship'' button on the UI (Fig. 11). They are created by using guiding lines to sweep TUBEs (one of UG primitives) with an API function. To represent a blind ''hole'', chamfer function is then called to add chamfer at the appropriate end. Cooling solids can be deleted by clicking ''Delete Cooling Channel'' button, but the types and parameters are still attached with the individual guiding lines. These attributes can further be displayed and edited with the same UI as shown in Fig. 11. Cooling solids can be regenerated anytime when the user wants.



p3 = 100
p4 = 50

Fig. 9. Repositioning sub-menu.

In more details, solid generation algorithms can be grouped into the following six functions, i.e. the creation of the following types of channels: simple blind, simple through, counter-bored blind, counter-bored at one end and through, counter-bored at the both ends and through, and finally, collinear cooling channels across multiple solids. The details of algorithms, including editing and deleting cooling channels, are not explained in this paper due to the space limit.

For each of the above functions, the user's input parameters and sequences are differentiated with the corresponding algorithm branches. A friendly UI is designed for each case. A few simple cases are explained in the following sections.

5.2.3.1. Creating simple blind "hole" solids. The simplest case is to create a simple blind hole based on the user-selected penetrating face. Based on the guiding line started from the penetration point, and the reverse normal vector of the penetrating face, a dynamically dragging guiding line is created. The end point of the guiding line is then a function of the penetration point, the drilling vector and the depth of the hole. Then the solid is generated with TUBE function and the end of the hole is chamfered according to the default angles and distances. The solid is associated to the penetration face as well because the guiding line is "tied" to it.

The second case in this algorithm is to create a blind "hole" solid based on a selected penetrating face and a perpendicular reference "hole". Fig. 12 shows the scenario. The moment a planar face is selected, the drilling vector is determined. The additional selection of another reference cooling solid serves to adjust the start point in *Y*-direction, so that, the new cooling-hole can intersect with the reference one. Note that in Fig. 12, if the user indicates point P initially, the system automatically set the start point to Q because the reference cooling solid.

In certain cases, although the drilling vector is determined when the user first selects the penetrating face, if the user selects another reference cooling solids, the drilling vector will be adjusted. In Fig. 13, it can be seen that if the user select cooling solid 1 as an addition reference after selecting the penetrating face, the drilling vector will be flipped from the initial direction to the opposite. Note that the user can always reverse the drilling vector through an UI button. Besides, in this case, when updating the drilling vector, the associated face needs to be changed too.

5.2.3.2. Creating simple through "hole" solids. This algorithm uses similar functions as creating simple blind hole, such as getting the start point and drilling vector, UI data retrieval, and data verification. Some minor modifications come from the reasoning for the through end of the "hole". When the exiting face is selected, based on the start point and the drilling vector, an infinite line is created. The intersection point of this line and the exiting face is made a smart end point.

5.2.3.3. Creating counter-bored blind or through "hole" solids. This algorithm makes use the same methods as the previous two cases except the creation of the counter-bore portion. It is generated as a TUBE again on top of the base cylinder with counter-bored outer circle profile for a given depth of it. This method is a function of the penetrating point, the drilling vector and the depth of the counter-bore. The

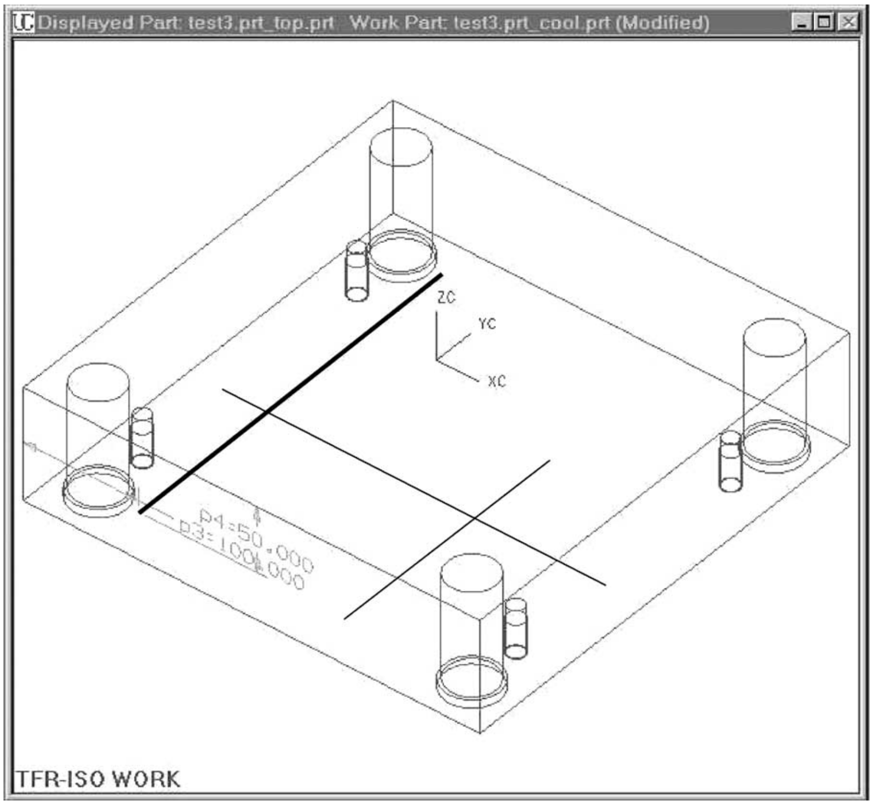

Displayed Part: test3.prt_top.prt Work Part: test3.prt_cool.prt (Modified)
ZC
YC
XC
p4=50.000
p3=100.000
TFR-ISO WORK

Fig. 10. Screen snap for editing guiding line position.

counter-bore portion is united with the base cylinder immediately to form the cooling solid.

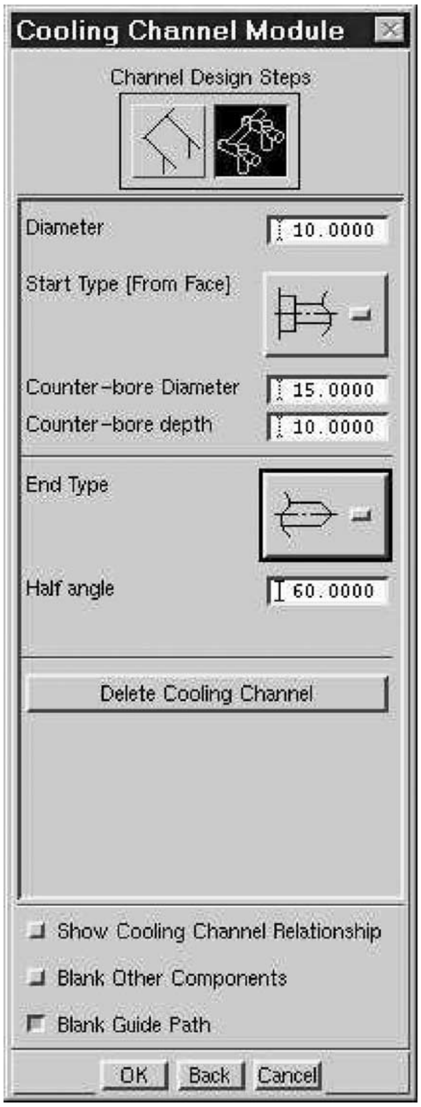
5.2.3.4. Creating collinear cooling channel. The objective to provide a method for the creation of a collinear cooling channel is to achieve the association among collinear individual holes. The key is that the start and end points of each hole are ‘‘tied’’ to an appropriate parent, such that when the parent is being modified, the child will be notified of the modification, and updated. Refer to Fig. 5 again for the illustration.

Assuming the first cooling-hole (Hole 1; from left to right) is created via ‘‘Create a counter-bored through hole by selecting two planar faces’’. Therefore, the start point of the hole A is ‘‘tied’’ to Face 1 and the end point B is ‘‘tied’’ to Face 2. Note Face 1 and Face 2 are part of Solid 1. Any modification to these faces, such as offsetting them, will affect the depth of the hole.

The creation of the middle hole has more flexibility. The user can create it with either the methods of ‘‘Create a counter-bored through hole by selecting two planar faces’’ or ‘‘Create a counter-bored through hole by selecting a cooling ‘hole’ and a planar face’’.

For the first case, Face 3 and Face 4 (belonging to solid 2) can be selected as the references for creating Hole 2. Hence, start point C and end point D are children of Face 3 and Face 4, respectively. Because this hole is supposed to be a part of the collinear channel, Face 2, which is associated with the end point (B) of Hole 1, is also made linked to Face 3. Because Face 3 is linked to the start point (C) of Hole 2, then it is associated to B in turn (see also Fig. 14a). This is assured by the ‘‘validator’’ of the collinear channel. Hence, the first ‘‘hole’’ can slide along Face 2 without upsetting the middle hole by thus creating two misalign holes.

In the second case, the parent of start point C is the end point of the first ‘‘hole’’, i.e. point B. If the first


Cooling Channel Module

Channel Design Steps

Diameter 10.0000

Start Type [From Face]

Counter-bore Diameter 15.0000

Counter-bore depth 10.0000

End Type

Half angle 60.0000

Delete Cooling Channel

☐ Show Cooling Channel Relationship

☐ Blank Other Components

☑ Blank Guide Path

OK Back Cancel

Fig. 11. The UI for cooling-hole end attributes input.

"hole" is modified by sliding Face 2, the middle "hole" will follow suit too. Due to the link between start point C of Hole 2 and Hole 1's end point B, once point C is moved, Face 3 will be updated too (see also Fig. 14b). This smart association between the two holes creates embedded relations among multiple solids with a collinear channel.

Similarly, the third blind hole from left to right in Fig. 5 can be created with these two options: (1) "create a counter-bored blind hole (one end) by selecting a cooling-hole" or, (2) "create a counter-bored blind hole (one end) by selecting a planar face". Hence, the end result is the collinear cooling channel consisting of three associated cooling-holes.

So far, we have discussed about creating a collinear channel from scratch. Sometimes, the user needs to create a collinear channel based on an existing reference cooling-hole (see Fig. 15). Such case is also supported. To calculate the start point of the next "hole", in this case, the type of the selected existing hole is important. In Fig. 15, as shown, the selected hole is a blind one, then the next "hole" start point can only exist at the point A instead of B, because the hole has a blind end at B. There is no logical collinear connection can be made at point B.

If the reference hole is a through one, then logically the intended hole can be started at either end of the selected hole. In Fig. 5, assuming Hole 2 is the existing reference, the user can create either Hole 1 from point C or Hole 3 from point D. To determine which one is the next, the closer end to the user's selecting point is used to create the next collinear hole.

5.2.4. Modifying cooling solids

Cooling solids can be modified anytime within the cooling channel module. When a cooling solid is selected for editing, the parameters and end types of the solid are displayed in an UI (shown in Fig. 11). These parameters and end feature types can be changed and updated. At the same time, the attributes of their corresponding guiding lines are updated.

5.2.5. Deleting cooling solids

Since cooling solids can be regenerated anytime with the guiding lines, therefore, for display purpose, the selected cooling solids can be deleted and regenerated with the guiding paths. However, if the user deletes the guiding path together, then the deleted cooling solids can no longer be regenerated.

5.3. Dealing with balanced and unbalanced cooling circuits

In a mold set, it can be designed to produce several moldings in a shot. Each molding cavity is called an impression. When producing a family of moldings, such as a toy set, the mold impressions are quite different. Therefore, the thermal distribution has to

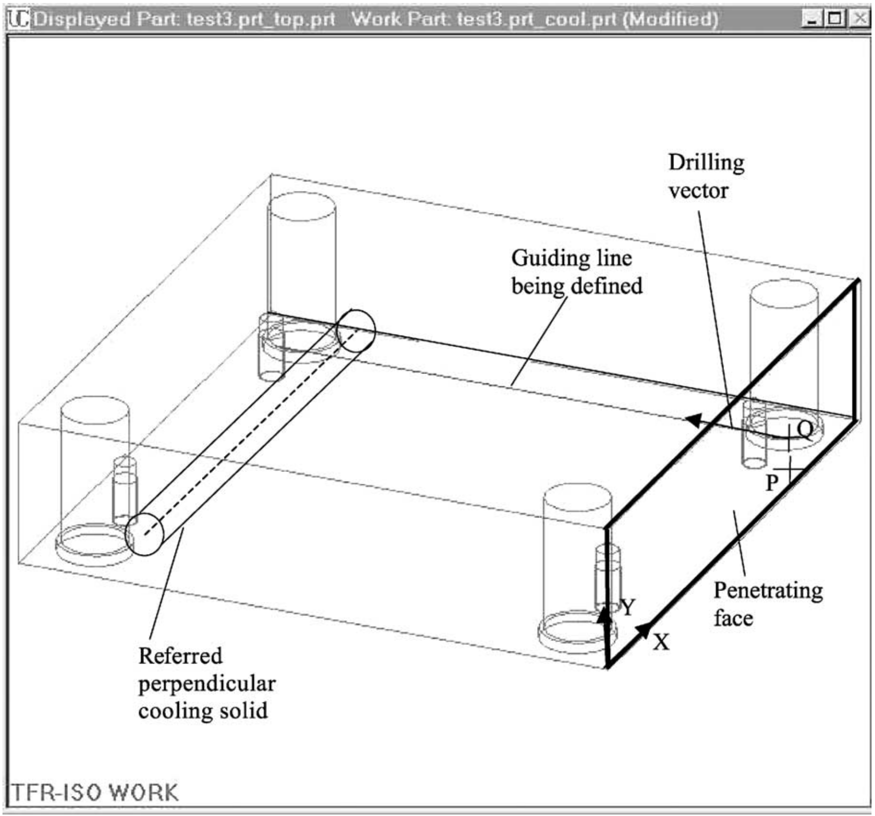

Displayed Part: test3.prt_top.prt Work Part: test3.prt_cool.prt (Modified)
Drilling vector
Guiding line being defined
Q
P
Penetrating face
Y
X
Referred perpendicular cooling solid
TFR-ISO WORK

Fig. 12. Define a cooling solid with the reference of an existing perpendicular cooling solid.

be considered from overall layout instead of individual impressions; in this case, the mold impressions are unbalanced. When producing identical moldings with several impressions, the thermal distribution problem can be simplified by considering a single impression only by assuming all other impressions have the same thermal pattern. Then, the mold can be designed with a symmetrical layout; in this case, the mold impressions are balanced.

Similarly, there are two methods to create cooling circuits: balanced and unbalanced. If the mold is designed with a balanced multi-impression pattern, and the designer wishes to have an identical cooling circuit for each impression, then the balanced method should be used. In this case, because each circuit is designed mainly to cover one impression, therefore the cooling effect can be better controlled to satisfy heat-transfer requirements. This is especially recommended for complex moldings where the cooling effect analysis has been carried out with some simulation packages [19,28].

In this work, all the mold assembly components are organized with a tree structure. This tree is automatically created when the user initializes a new mold design project. The original plastic part is assigned under the top-assembly part and is referred to as the Product Part (Prod-part) (refer to Fig. 16). Another part specially designated for cooling solids is created under the top-assembly as well; it is named as the Cooling Line part. This part is automatically created in the predefined-assembly structure when the cooling channel module is activated for the first time.

In order to address this balanced and unbalanced cooling circuit design issue, two UG concepts, waved entity and work part, have to be introduced first. UG has a special technology, WAVE, that enables certain geometrical entities to be referred to associatively with certain functions, such as COPY and TRIM,

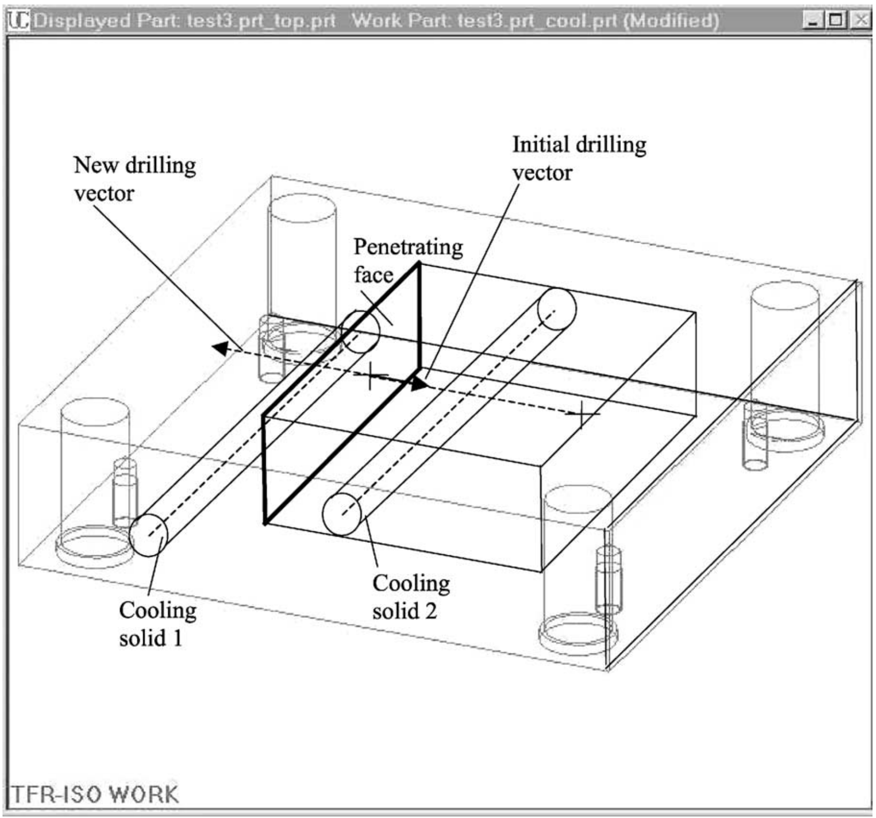


Fig. 13. Alternation of the drilling vector when selecting a penetration face with two different reference cooling solids.

among different parts in an assembly. This is achieved by copying the entities from one part to another with persistent association. Those entities that are instantiated in the destination part by copying from the original entities are referred to as *waved entities*. They share all the properties of, and are persistently associated to, the source entities. It means when a source entity is modified, its corresponding waved entity gets updated automatically as well. Two examples of possible waved faces in an assembly are shown in Fig. 17.

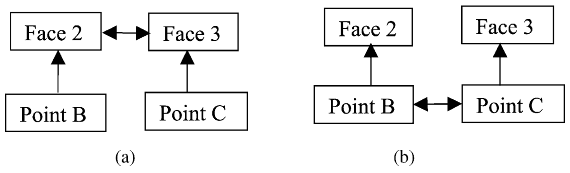


Fig. 14. Two cases to interface two cooling-holes in a collinear channel.

Assume a source face is in component part 1, it can be waved to create an associative copy, Face 1 in its parent part (child-to-parent); or Face 2 in component part 2 (child-to-child). For the second concept, in UG assembly modeling environment, the *work part* is the part where new entities are created.

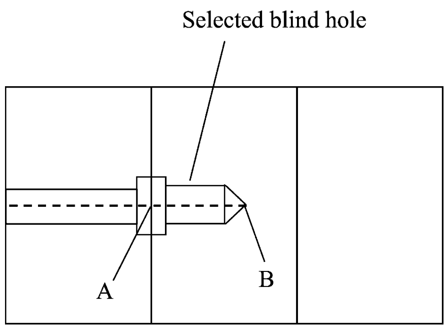


Fig. 15. Creating a collinear channel with an existing blind hole.

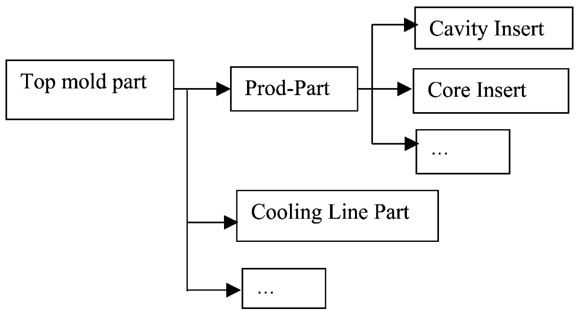


Fig. 16. Cooling line part in the mold assembly tree.

Balanced impression design is achieved by applying a component pattern on Prod-part together with the core/cavity inserts within UG top-assembly structure. In UG, component pattern function can only be applied to an immediate child (or sub-assembly) of an assembly. The idea of this work is, when creating balanced cooling circuits, to create cooling entities in the prototype Prod-part, so that they can be instantiated together with the core/cavity inserts in the same pattern. Therefore, the work part is set to Prod-part (see Fig. 16). When the user selects a face in core/cavity, a waved face is created in Prod-part (child-to-parent). Then, the master cooling elements, including smart points, guide paths and cooling solids, are created in Prod-part. Hence, when applying the component pattern for different impressions, all instances are updated with the same cooling circuit. At the same time, waved guide paths and solids are created in the CL part (child-to-child); hence, with this part, cooling entities can be easily managed and manipulated transparently. In Fig. 18, some balanced cooling circuits are illustrated.

When creating unbalanced cooling channels, the work part is set to the CL part. When user selects a face in core/cavity inserts, a waved face will be created in Prod-part (child-to-parent) and then in the CL part (child-to-child). However all related cooling entities, such as smart points, guide paths and cooling solids, are created in the CL part only. Prod-part and CL part are then associated such that the cooling entities in the CL part will be updated automatically if their linked entities in Prod-part are changed. In other words, when modifications occur in mold insert parts, the cooling line part will also be refreshed automatically.

It can be appreciated that for the both cases, the assembly tree structure enables the effort for design rework, due to late product changes, largely reduced and hence the efficiency enhanced.

5.4. Logical methods—“creators” and “validators”

In this cooling design module, there are two types of logical methods of importance, “creators” and “validators” (see Fig. 2). They can be viewed as logical methods because they keep an object to be self-contained and well-defined. “Creators” are the methods to construct the corresponding object contents. For example, the “creator” of cooling guiding line is to create an instance object with different input combinations and to validate whether a selected face can be the penetrating face for a hole. The role of cooling-hole “creator” is to subtract cooling solid from their corresponding plates or inserts and hence to create cooling-holes. This type of methods decides if relations between objects can and should be maintained

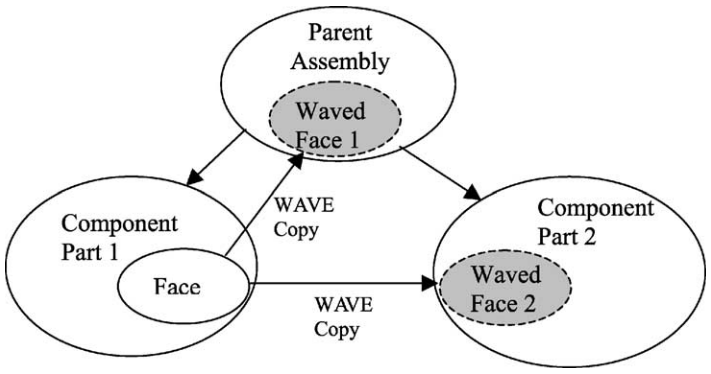


Fig. 17. Two examples of possible waved faces in an assembly.

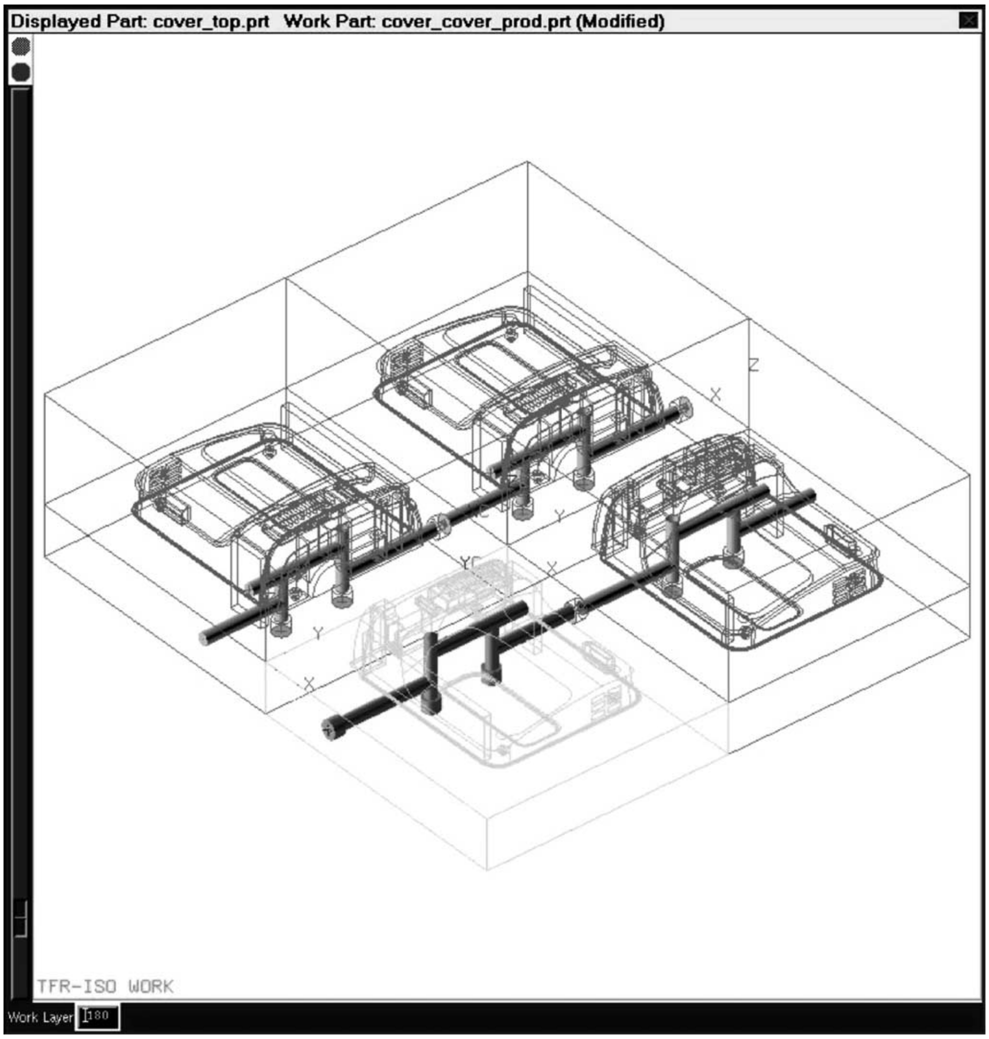

Displayed Part: cover_top.prt Work Part: cover_cover_prod.prt (Modified)
TFR-ISO WORK
Work Layer 180
X
Y
Z
YC

Fig. 18. An example of balanced cooling circuits.

and builds links across all associated cooling elements. "Validators" are designed to verify the integrity and validity of objects. Three different "validators" are embedded in guiding path, cooling solid and collinear channel objects, respectively. "Validators" verify the inputs of users, and invoke the next process if the inputs are acceptable, or provide feedback to the user if necessary. They ensure guiding lines/path connectivity as well as legal user inputs. Clearly, the functionality of these logical methods can be further expanded. Many more logical rules can be implemented.

5.5. Interactions

In order to avoid over complex display for mold designers, cooling smart points, guide paths, and solids in the Prod-part are put into some specific layer and set to specific colors. This layer is set to be visible when initializing Cooling Channel module and invisible when quitting Cooling Channel module. Clearly, in the cooling channel module, many user interaction scenarios are involved. They cannot be extensively described due to the space limitation.

6. Potential integration with other applications

With the object definition of a cooling system, conceptual cooling design can be simplified with the help of cooling guiding line creation methods. Detailed parameters and solid representation can be left undefined until the design is ready for further steps. Cooling circuit patterns can also be incorporated easily with step-by-step enrichment of attributes and the selection of associated faces.

Technically, if the cooling channels are modeled in this associative feature form, CAE analysis for cooling effect evaluation can be easily integrated because the cooling guiding path can be used as the circuit mesh [19]. For example, those logical rules that are connected with mold design CAE analysis in [25,27,28] can be implemented with certain query and execution methods. The authors believe that this should be the coming research direction, and it can be categorized under a topic, i.e. knowledge driven automation (KDA). The cooling effect can be further measured in the form of another object. It can be updated every time after the change of the cooling system by running the analysis methods again and again.

Similarly, collision check can be carried out automatically by generating the cooling solids, and checking them against other features or components with a cycling algorithm. Most importantly, such integrated applications can be achieved by making use the methods defined with the object of the associative cooling system. Those "validators" introduced above assure the consistency of the cooling system.

On the other hand, if the designer makes any changes over the cooling system, it can send notifications to other related applications, or even trigger the analysis functions automatically. It is also obvious that when mold plates and inserts are modified, cooling channels will be updated automatically. This function provides timely feedback to check if any constraint is violated. The output solid after Boolean operations between the housing blocks and the cooling channel solids can be directly used for CAM processing.

With the information of hole-positions, drilling direction and the penetrating and exiting faces, CAM machining cycles can be called for CNC codes generation. In assembly modeling, associated components like hose nipples and plugs, can be added with "drag-and-drop" manner. They can also be updated when cooling channel positions or orientations are changed.

7. Conclusion

This paper has described an object-oriented associative design approach and proposed a new feature type, namely, associative features. Associative cooling channels in injection molds have been implemented and used to illustrate the concept, methods and the run-time behavior. It gives a comprehensive object definition of cooling circuits. With this new approach, mold designers can easily propagate changes from mold plates or inserts to the cooling system without tedious rework. It also supports integration with other engineering applications, such as conceptual design and CAE analysis. Therefore, it enables design-analysis-redesign cycles with a consistent data structure. The emphasis is put on the built-in geometrical associations among cooling channels and mold plates or inserts for feature integrity.

Acknowledgements

Unigraphics^{TM} (UG) and MoldWizard^{TM} are registered trademarks of EDS Inc. The authors would like to acknowledge that the research work presented here is based on the teamwork of CAPT group of Singapore Institute of Manufacturing Technology. EDS Inc., Cypress, USA, had provided close technical support. This paper is intended to report a new research approach; it does not reflect the current and future versions of MoldWizard^{TM}.

References

[1] B. Prasad, Concurrent Engineering Fundamentals: Integrated Product and Process Organization, Prentice Hall, Englewood Cliffs, NJ, 1996.

[2] P. Gu, K. Chan, Product modeling using STEP, Computer Aided Design 27 (3) (1995) 163–179.

[3] M. Hardwick, D.L. Spooner, T. Rando, K.C. Morris, Sharing manufacturing information in virtual enterprises, Communication of the ACM 39 (2) (1996) 46–54.

[4] F.E. Brown, G.S. Cooper, S. Ford, G. Aouad, P. Brandon, T. Child, J.A. Kirkham, R. Oxman, B. Young, An intelligent

approach to CAD: modelling concepts in building design and construction, Design Studies 16 (1995) 327–347.

[5] CASA/SME Board of Advisors, Virtual enterprise integration: creating a sustainable manufacturing life cycle, CASA/SME Blue Book Series 2001; website: http://www.sme.org/casa.

[6] S.K. Ong, S. Prombanpong, K.S. Lee, An object-oriented approach to computer-aided design of a plastic injection mold, Journal of Intelligent Manufacturing 6 (1995) 1–10.

[7] D. Xue, S. Yadav, D.H. Norrie, Knowledge base and database representation for intelligent concurrent design, Computer Aided Design 31 (1999) 131–145.

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

[9] N. Kishi, An expert system tool for an automotive design, International Journal of Vehicle Design 11 (3) (1990) 272–280.

[10] I.B.H. Lee, B.S. Lim, A.Y.C. Nee, IKOOPPS: an intelligent knowledge-based object-oriented process planning system for the manufacture of progressive dies, Expert Systems 8 (1) (1991) 19–33.

[11] O.W. Salomons, F.J.A.M. van Houten, H.J.J. Kals, Review of research in feature-based design, Journal of Manufacturing Systems 12 (2) (1993) 113–132.

[12] H. Yoshikawa, T. Tomiyama, T. Kiriyama, Y. Umeda, An integrated modeling environment using the metamodel, Annals of the CIRP 43 (1) (1994) 121–124.

[13] U. Roy, Y. Xu, L. Wang, Development of an intelligent inspection planning system in an object oriented programming environment, Computer Integrated Manufacturing Systems 7 (4) (1994) 240–246.

[14] J.J. Shah, M. Mäntylä, Parametric and Feature-based CAD/CAM, Wiley, New York, 1995.

[15] K. Case, Using a design by features CAD system for process capability modeling, Computer Integrated Manufacturing Systems 7 (4) (1994) 39–49.

[16] H.E. Otto, From concepts to consistent object specifications: Translation of a domain-oriented feature framework into practice, Journal of Computer Science and Technology 16 (3) (2001) 208–230.

[17] M.-W. Fu, J.Y.H. Fuh, A.Y.C. Nee, Core and cavity generation method in injection mould design, International Journal of Products Research 39 (1) (2001) 121–138.

[18] Y.-S. Ma, Y.-Q. Lu, Z. Li, Associative design and knowledge automation in MoldWizard^{TM}, in: Proceedings of the First International Conference on Information Technology and Application (ICITA 2002), Bathurst, Australia, 25–28 November 2002.

[19] K. Himasekhar, J. Lottey, K.K. Wang, CAE of Mold Cooling in injection molding using a three-dimensional numerical simulation, Journal of Engineering for Industry 114 (1992) 213–221.

[20] Y.-S. Ma, T. Tong, An object-oriented design tool for associative cooling channels in plastic injection mold, International Journal of Advanced Manufacturing Technology.

[21] S.K. Ong, A.Y.C. Nee, Application of fuzzy set theory to setup planning, Annals of the CIRP 43 (1) (1994) 137–144.

[22] R.G. Pye, Injection Mold Design, third ed., Longman, New York, 1983.

[23] J. Cunningham, J. Dixon, Designing with features: the origin of features, in: Proceedings of ASME Computers in Engineering Conference, San Francisco, USA, July/August 1988.

[24] C.L. Li, A feature-based approach to injection mould cooling system design, Computer Aided Design 33 (2001) 1073–1090.

[25] K.J. Singh, Design of mold cooling system, in: A.I. Isayev (Ed.), Injection and Compression Molding Fundamentals, Marcel Dekker, New York, 1987, pp. 567–605.

[26] R.K. Irani, B.H. Kim, J.R. Dixon, Towards automated design of the feed system of injection molds by integrating CAE, iterative redesign and features, Journal of Engineering for Industry, Transactions of the ASME 117 (1995) 72–77.

[27] T.H. Kwon, P.A. Weeks, Expert system aid for intelligent molding cooling system design, in: Proceedings of the 1988 Computers in Engineering Conference, ASME, San Francisco, CA, USA, vol. 1, 1988, pp. 281–286.

[28] K.K. Wang, V.W. Wang, Computer-aided mold design and manufacturing, in: Avraam I. Isayev (Ed.), Injection and Compression Molding Fundamentals, Marcel Dekker, 1987, pp. 607–669.

[29] C.K. Mok, K.S. Chin, J.K.L. Ho, An interactive knowledge-based cad system for mould design in injection moulding processes, The International Journal of Advanced Manufacturing Technology 17 (2001) 27–38.

[30] T. Kishinami, T. Kikuchi, S. Kanai, K. Saito, Development of interactive mould cavity CAD/CAM system, Annals CIRP 32 (1) (1983) 395–399.

[31] S.W. Lye, H.Y. Yeong, Computer-assisted mould design for Styrofoam products, Computers in Industry 18 (1992) 117–122.

[32] G. Salloum, D. Charland, X. Brouwers, PLASFEED—an interactive design software of feeding system for injection mold using solid modeling, in: Proceedings of the 44th SPE Annual Technical Conference and Exhibition (ANTEC'86), Boston, 1986, pp. 1412–1418.

[33] W.R. Jong, K.K. Wang, Automatic and optimal design of runner system in injection molding based on flow simulation, in: Proceedings of the 48th SPE Annual Technical Conference and Exhibition (ANTEC'90), Dallas, 1990, pp. 385–389.

[34] G.A. Britton, Y.-S. Ma, S.B. Tor, Object technology development and unigraphics, in: Proceedings of Unigraphics User Group 1999 Spring Conference: Managing Design Evolution, New Beach, CA, USA, 1999.

[35] X.-G. Ye, J.Y.H. Fuh, K.S. Lee, Automated assembly modeling for plastic injection moulds, The International Journal of Advanced Manufacturing Technology 16 (2000) 739–747.

[36] EDS Inc., User guide for unigraphics v16 (user functions), UG Software Document.

Dr. Y.-S. Ma is currently an associate professor at School of Mechanical and Production Engineering, Nanyang Technological University (NTU), Singapore. His main research areas include intelligent modelling and object technology for engineering IT solutions. Graduated from Tsing Hua University, Beijing with BE degree in 1986, he further studied at UMIST, UK and achieved his Msc and PhD degrees in 1990 and 1994, respectively. He started his career as a lecturer at Ngee Ann Polytechnic in Singapore from 1993 to 1996. From 1996 to 2000, he worked in Singapore Institute of Manufacturing Technology, as a Research Fellow, Project Leader, Senior Research Fellow and Group Manager. He joined NTU since September 2000.

Mr. T. Tong is currently a PhD candidate at CAD/CAM Laboratory, School of Mechanical and Production Engineering, Nanyang Technological University, Singapore. He graduated from Shanghai Jiao Tong University and received BE degree in 1998. From 1998 to 2001, he worked as a Project Engineer in Shanghai HITACHI Electrical Appliances Co. He entered NTU in August 2001.