Associative Design and Knowledge Automation in MouldWizard

Y. –S Ma, Y. –Q Lu, and Z. Li

***Abstract*-- MoldWizard™ is a software product of Unigraphics Solutions Inc. It offers a special process based solution for plastic injection mold design. This paper describes the tooling design knowledge encapsulated in four modules of MoldWizard™, i.e. Cooling Channels, Gate and Runner, Electrode and Sub-Insert. Conceptual ideas are briefly explained.**

***Index Terms*—Design Automation, CADCAM, Manufacturing, and Knowledge Engineering**

I. INTRODUCTION

ACCORDING to the definition of SME [SME1999], CAD interoperability should cover the relationship between a KBE system and a CAD platform. However, to transfer information from CAD to knowledge-based Engineering (KBE) systems is very difficult because KBE systems rely heavily on the design intent to perform activities such as cost estimating 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, transfering KBE intelligence to CAD systems is equally challenging because there is no mechanism to enable such information flow. One way to bridge these gaps is to build design intent into the CAD system in the form of process wizards. Basically, a wizard is a set of sequenced UI interfaces to guide the users to complete certain interactions. Wizard tools can bridge the gap between human experiences and computer based activities. MoldWizard™ is such a wizard for plastic mould design. This paper describes the conceptual design for four modules, i.e. Cooling Channels, Gate and Runner, Electrode and Sub-Insert.

II. COOLING CHANNELS

In current industry practice, straight solid cylinders are created to represent cooling channels. In the cases of blind channels, the cylinders are chamfered at the blind ends to make them appear as drilled blind holes. When the design is finalized, all channels are united to form a cooling circuit. The reasons for using solids instead of "hole" features are as follows:

- The creation of cooling holes/channel entails several time-consuming, manual tasks;
- Solids representation enables cooling circuit drawing without cavity or core block, mold plates, etc.
- Repositioning holes requires many more steps.

However, using native Unigraphics Modeling functions to create cooling solid cylinders has some shortcomings. Except a number of steps required, no intelligent representations for cooling channels. For example, cooling cylinders cannot be identified specifically. This is important because the cooling channels need to be used for thermal analysis purpose. There is also no orientation and connectivity information Therefore, there is a need to have a customized module for designing cooling channels. The aim of this system is to provide substantial automation and intelligence (smart association) in the process of cooling solids generation.

A. Definitions

In MoldWizard™, the definition of cooling circuit is a set of holes, which are connected to each other with an inlet and an outlet. A cooling circuit may have several holes but can also be only one hole. Some common cooling “hole” terminology is shown in Fig. 1. In the CAD model, holes are represented initially with their centerlines with special attributes to specify the hole parameters. For every cooling hole, its base point and tip point are made associated with penetrating faces except the blind tip point. In the Unigraphics™ terms, they are "smart points". Solids are then generated by the program and united as the volume representation of cooling circuits. A cooling solid is the tub solid created by sweeping along a guiding line, which is a “smart” line because it connects two smart points. Hence, in turn, base on this smart line, the tub cylinder is “smart” too due to Unigraphics reference mechanism. All the guiding lines within a cooling circuit form a guide path. Different “hole” types are defined in Fig. 2. Related attributes need to be assigned to the guiding lines. They are used to regenerate “hole” solids when never necessary. Within MoldWizard™, all the mold assembly components are organized with the UG assembly tree structure. By default, a cooling line (CL) component is specially created for cooling solids under the mould assembly.

MoldWizard™ and Unigraphics™ are products of Unigraphics Solutions Inc. This work describes the conceptual frame only. The four modules introduced here were jointed developed by Gintic Institute of Manufacturing Technology (Gintic) in Singapore and Unigraphics Solutions Inc., USA.

Y. -S. Ma (telephone: 65-6790-5913, e-mail: mysma@ ntu.edu.sg), is currently an associate professors, school of MPE, NTU, 50 Nanyang Avenue, Singapore 639798.

Y. -Q. Lu, is a group manager of Gintic Institute of Manufacturing Technology (Gintic), 71 Nanyang Drive, Singapore 638075.

Z. Li is an R & D project leader of Unigraphics Solutions Inc, 10824 Hope Street, Cypress, CA 90630, USA.

** LEAVE TWO LINES HERE FOR IEEE TO ADD THE PUBLICATION NUMBER AND OTHER CONFERENCE INFORMATION.

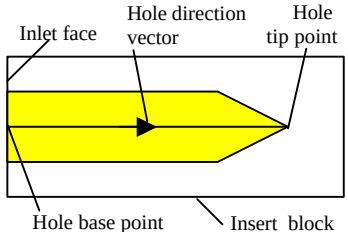


Fig. 1 Some Common Cooling Hole Terminology

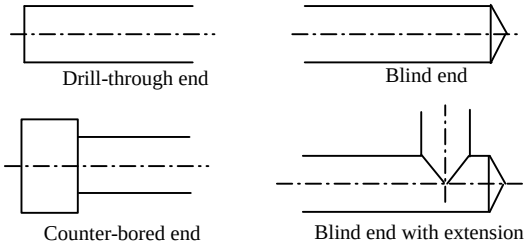


Fig. 2 Types of cooling hole ends

B. *Functional Design*

Cooling line module provides the following functions with the association of a cooling circuit with related faces:

- Creating cooling circuit with smart guide path,
- Adding /Removing curve from guide path,
- Modification/Reposition of guide path,
- Deleting of cooling circuit guide path,
- Creation of cooling solid,
- Modification of the cooling solid, and
- Deleting of the cooling solid.

To create the first guiding line of the guide path, the user needs to select a face as the inlet penetrating face of the circuit. A smart point will be created on the face. The default drilling direction (the direction to generate the first cooling guiding line) will be set to the reverse direction of the face normal. The user can flip the direction if he wants.

The user can dynamically drag the guiding line, or input the length, or indicate another face for a through hole. For a through hole, another smart point will be created since the guiding line ends at the selected face. Guiding lines are sequenced. After creating one guiding line, 5 directions can be selected for the next one: +X, -X, +Y, -Y, +Z, -Z and User Defined. If there is no intersection between the guide path and the drilling face, the guide path will be extended automatically along the reverse direction of the indicated direction to find the drilling point on the face. Please refer to Fig. 3.

The designer may use balance circuits if the mold is designed with a balanced multi-cavity pattern, so that a separate circuit is used for each cavity section. Otherwise, unbalance ones may be used. In the balance design, when the user selects a face in core/cavity insert, a waved face is created in the product part. Smart points, guide paths and cooling solids are created in the product part too while the related waved guide paths and solids are created in CL part. Hence, the same cooling circuit can be copied by using the CL component pattern in UG assembly. In the unbalance design, when user selects a face in core/cavity, a waved face will be created in the CL part, and smart objects, guide paths and cooling solids will be created in the same part.

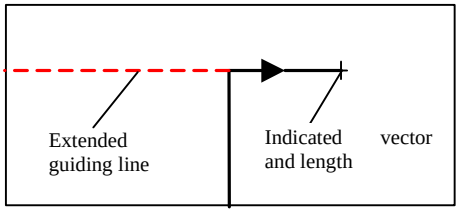


Fig. 3 Extension of the input line

C. *Design options*

The algorithms can be grouped into the following six groups, i.e. the creation of simple blind hole, simple through hole, counter-bore blind hole, counter-bore through hole at one end, counter-bore through hole at the both ends, and finally, multiple solids cooling channel. Other algorithms include editing and deleting cooling lines. They are not explained in this paper due to limited space. The user's input sequences are differentiated with corresponding algorithm branches. For example, for creating a simple blind hole, user's selection sequences can be any of the three options, i.e. (1) just the inlet penetrating face, or (2) the inlet penetrating face and then an existing perpendicular reference cooling hole, or (3) simply an exist cooling hole collinear to the intended one. Take an example, for the second case, after getting the inlet vector, the additional selection of another cooling channel only serves to adjust the inlet point in either the y or z direction (see Fig. 4 from point F to E). Although an inlet vector has already been determined when the user first selects the inlet face, the vector position is adjusted when the user picks the reference cooling hole.

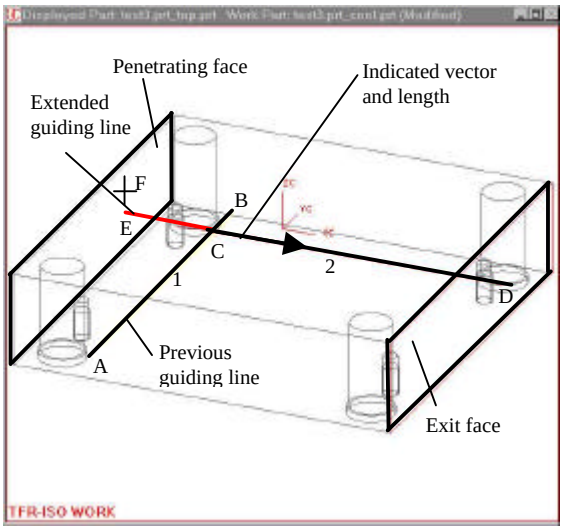


Fig. 4 Alteration of Inlet Position via Cooling Hole Selection

The values in the default cooling line data file are updated with the user's choice of preferred initial values when he/she quit the cooling dialog. Individual entries to different fields of the UIs are verified against preset conditions. For example, consider the pre-condition that the counter-bore diameter must be greater than the respective hole diameter.

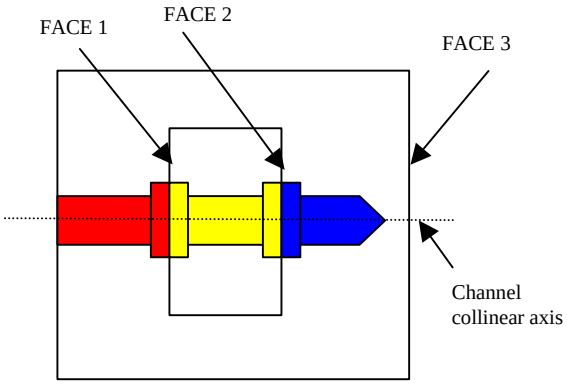


Fig. 5 A Typical Multiple Solid Cooling Channel

A special feature implemented is a method to create a multiple solids cooling channel (see Fig. 5) to achieve the association among collinear individual holes. In order to achieve this, the start and tip points of each hole is "tied" to an appropriate parent. The result of this is that when the parent is being modified, the child will be notified of the modification. Fig. 5 shows the association concepts. Assuming the first cooling hole (middle one) is created via "Create a counter-bore through hole with two faces, the base point of the hole is "tied" to Face_1 and the tip point of the hole is "tied" to Face 2. Any modification to, such as offsetting, these faces will affect the depth of the hole. The creation of the left counter-bore through hole (Both Ends) has more flexibility. The user can create it via two planar faces (Face 1 and the most left face) or via a cooling hole and a planar face (the middle hole and the most left face). The difference between the two methods is that the parent of the left hole base point is Face 1 in the first instance and the tip point of the first hole in the second case. In the first case, the first hole can slide along face_1 without affecting the left hole, by thus creating 2 misalign holes. In the second case, if the first hole is made to slide along the same face, the middle hole will follow suit too. This association between the two holes creates a multiple solids channel. Similarly, the third blind hole (right side) can be created via the middle hole too. The end result is a cooling channel consisting of three associated cooling holes.

III. Gate and Runner Module

In current industry practice, there is no consistent approach to create gate models by mold designers. The gates modeled are non-parametric. If the gates have to be modified due to product modifications, the gate models have to be completely re-built. In the case of multiple-cavity molds, the gates are usually modeled individually. Very often, when modification is required, the effort is tedious and cumbersome. The creation of the runner system involves the modeling the primary runner, branch runners and cold slug wells. Using native Unigraphics™ applications, this could involve the creation of guide strings (curves), cross-sectional curves and a host of other features. A substantial number of interactive operations are needed to create a complete runner design, especially in the case of multiple-cavity molds. Therefore, there is a need for a software module to enable mold designers to expedite the process of gate and runner creation and modification.

A. Assembly Structure

There are 3 possible structures to organize gates: (1) All gates are parametric solids and are accommodated in a single gate-runner component under the top mould assembly. With this structure, gates can be retrieved with three steps, i.e. detecting the layout, calculating the matrix and importing the gate part from library several times according to indicated position and the matrix. The advantage of this structure is the simple assembly tree. However, gates cannot be associated to any smart objects because of the constraint of importing part. Reference point is fixed values rather than an object pointer. When the layout changes, all gates do not move accordingly. All gates are individual solids without any link. Although it is possible to modify all gates in a batch within the Gate/Runner module, but when the user modifies a gate through native UG functions out of the module, the related gates cannot be updated. (2) All gates are components under the gate/runner sub-assembly. The tree structure is shown in Fig. 6.

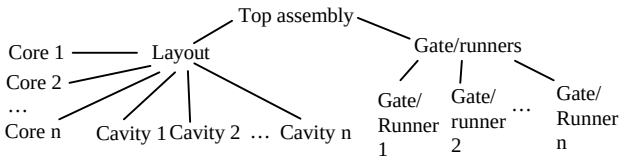


Fig. 6 Assembly structure of option 2

Implementation can be carried out by detecting the layout first and calculating the matrix. Program then add all gate/runner components as instances according to indicated position and the matrix. Since the gate/runner components are instances, if the user modifies a gate through native UG function, the related gates update accordingly. The disadvantage is when there are a lot of components in multi-cavities case, since all gate/runner components are not associated to any smart object due to the same reason as described in Option 1. When the layout changes, all of them become mismatched. When one of them is re-positioned, the related ones do not move. (3) Balanced gate/runners are stored as components under core/cavity sub-assemblies, unbalanced gates are stored as components under gate/runner sub-assembly (see Fig. 7). Waved solids are formed in the Gate/runner component for the display and selection purpose. The advantages for this option are numerous. When the gate/runners are balanced, if the user modifies one with native UG function, the related gates update accordingly. The

structure is not complex, as generally there are not many unbalanced gates. When a balanced gate/runner is re-positioned by either re-positioning, the related gates move. When the layout changes, all gate/runners move accordingly. In the multi-gate cases, the association between gates can be achieved. (In Fig. 7, "gate/runner 2.2" … "gate/runner 2.m" indicate instances of "gate/runner 2.1"). Base on above comparison, option (3) is chosen. All gates and runner channels generated by this module are solid bodies. The user can subtract these solid bodies from the core and cavity inserts with boolean functions.

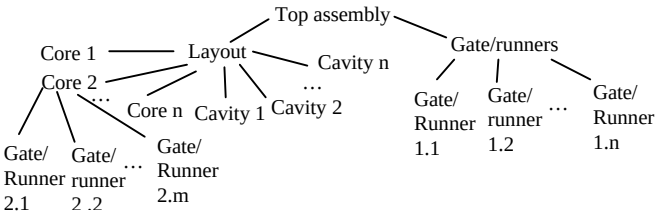


Fig. 7 Assembly structure of option 3

As to gates, based on the above functional requirements, a gate library has been created in MoldWizard™. The following pre-defined gate types are included in the gate library: rectangular, fan, submarine, pin, film and stepped pin. There is an option for users to choose between a balanced gate pattern and an unbalanced gate pattern. If the user chooses a balanced pattern, the gates in all cavities of the layout will be generated at the same time. The positioning of gates makes use of the associative method used by the Standard Parts module of MoldWizard™. The user selects a transformation method and an association option for the positioning. A new gate name has to be specified if non-associative copy is chosen.

For runner creation, some definitions are needed. Guide string refers to the curves along which runner solids are generated by sweeping the runner cross-section profiles. Guide pattern refers to the sketch defining the runner channel patterns. Usually, runner guide strings are created using Unigraphics lines and arcs. In this work, commonly used guide string patterns are made available (see Fig. 8); hence it is very efficient for multiple-cavity molds. Runner channels can be generated using specified cross-sectional topology (see Fig. 9) along with the guide strings. All the pre-defined cross-sections are parametric and the user can easily modify the sections via a UI dialog. Similarly, the user can select a pre-defined guide pattern from a standard library and specifies the necessary parameters through a UI dialog (see Fig. 10).

IV. Electrode Module

In mould making industry, using EDM to machine cavity is a common practice. To create an electrode, the following steps will be involved: (a) Creating an inverse shape of a portion or the whole of a mold impression component (i.e. core insert, integer core, cavity insert, integer cavity and sub-inserts, including certain types of gates); (b) Adding a base to the inverse shape (in practice, the base is tightened to a holder and the latter is fixed to the CNC machine when machining the electrode as well as the EDM machine when use it to make the core/cavity); (c) Adding a reference coordinate system to the electrode for machining purposes, and (d) Adding other reference features, such as chamfers, to the base to indicate the front side so that the electrode is positioned correctly during the EDM process. The above generally involves a substantial number of steps and the aim of this module is to reduce the effort required by the end-user in the electrode generation process.

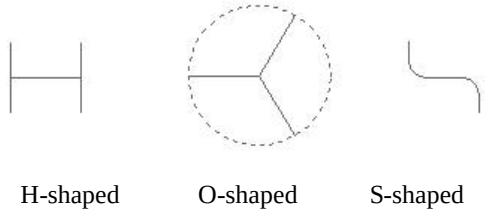


Fig. 8 Typical runner channel guide string types

Fig. 9 Typical runner cross-section profiles

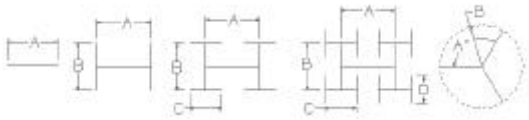


Fig. 10 Implemented runner patterns

A. Definitions

An electrode consists of mainly two portions, i.e. the electrode head and the base. The electrode of any portion of the cavity is represented as an associative solid and stored in a designated sub-assembly. The assembly structure is created as illustrated in Fig. 11.

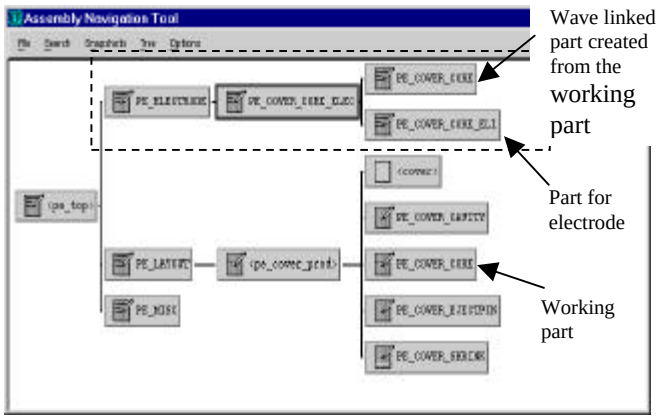


Fig. 11 Assembly structure for the electrode module

The working part in the above assembly structure is "PE_COVER_CORE". An sub-assembly "PE_ELECTRODE" is added to "pe_top" to contain all electrodes. Its components are named with the suffix "_ELEC" appended to the original working part name, e.g. "PE_COVER_CORE_ELEC". A

UG/WAVE[3] linked original part (with the same name) is added to this node too. The display part is then set to the electrode parent part (“ PE_COVER_CORE_ELEC”) so that user can see and select geometry from the UG/WAVE linked part of the working part. A separate part is created for every electrode and each is set as the working part when the electrode is being created. The rationale of using the assembly structure described above is to avoid copying all the geometry of the working part into the electrode part, which would occupy a lot of space. Only when needed is some geometry from the working part copied with UG/WAVE links to the electrode part. For example, if the user decides to create the electrode head by a trimming operation form the parting sheet, then only the parting sheet is wave-copied to electrode part.

B. Functional Description

In order to serve the requirements for creating electrodes, the module provides facilities to perform the following functions (and sub-functions, where appropriate): creation of enveloping block for electrode, creation of electrode head, editing of electrode head, creation of electrode base, editing of electrode base, and creation of reference coordinate system at highest point of electrode for EDM/CNC setup.

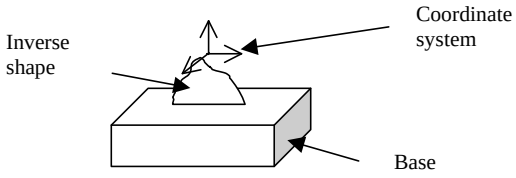


Fig. 12 An electrode diagram

A simple diagram is shown in Fig. 2. The above functions are briefly described individually below. It is assumed that the cross-section of the electrode head takes the shape of either a circle or a rectangle when viewed from the top of the main core or cavity insert. Interactively, a sketch is built on a datum, which is offset by an adjustable distance from the core/cavity. After the user confirms the dimensions of the sketch, the sketch is extruded in opposite directions to form the enveloping block. Functions have been built to help user to position and dimension the sketch.

An electrode head should be automatically created with face taking from the impression at a specific location within the main core or cavity insert. The electrode head can be generated by either of the following two methods, as specified by the user. In the first method, the enveloping block is trimmed by the large trimming sheet that has been generated by the parting process. In the second method, where the large trimming sheet does not exist, the core/cavity is copied, and this copy is subtracted from the enveloping block.

When a user moves the positions and sizes of the sketch, the electrode head will be modified accordingly.

A function has been available is to create the electrode base (see Fig. 13) with the electrode block (either a solid rectangular block or a cylinder), in a position and orientation specified with reference to the electrode head. A separate block representing the electrode base, with two chamfers at its corners to indicate its front orientation is automatically created. An offset face of the electrode head may also be created if tool wearing has to be considered. The user inputs include a reference face on the electrode block, the base normal direction, the position of its lower face, and its dimensions. Of course, the electrode base can be repositioned or resized.

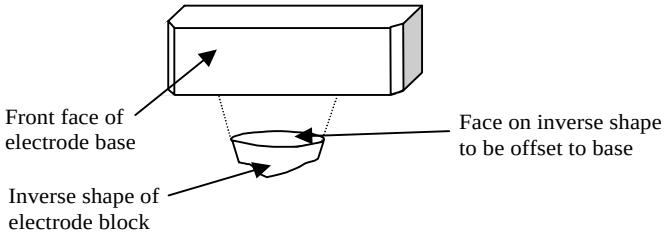


Fig. 13 Rectangular electrode base

For CNC or EDM origin setting up purpose, a coordinate system at the highest point of the electrode block to be used as a reference point for machining purposes. The coordinate system will be horizontally positioned at the midpoint of the electrode block’s bounding box (see Fig. 14).

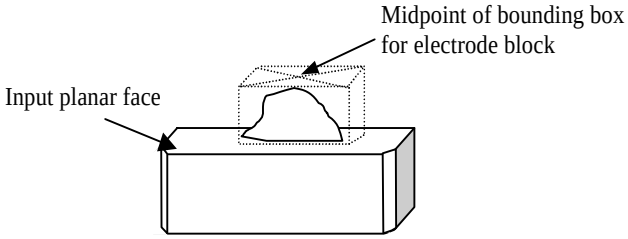


Fig. 14 Reference point for an electrode

V. SUB-INSERT MODULE

Normally, sub-inserts are created after the creation of the main core and cavity inserts. The purpose to create sub-insert is to simplify the impression machining and to reduce the cost. In other words, a portion of the impression’s profile is used to create the sub-insert head. The process of creating the entire sub-insert can be divided into the following tasks: Creation of sub-insert head, Creation of sub-insert body, Creation of positioning and orientation features, and Creation of fastening features.

A sub-insert usually has a body to hold the head. In some cases, however, sub-insert bodies are not created and the sub-insert head is directly mounted onto a mold plate. In all cases, the generation of the entire sub-insert shall not modify any existing solids. The sub-insert will be created as separate solids. This module is very similar to the electrode module except that the sub-insert head does not need to use the reversed impression face. Therefore, due to the paper space limit, this module is not elaborated here.

VI. Interaction

Clearly, in each of the module, many user interaction scenarios are involved. UI design is critical for wizard application. They cannot be extensively described here. Some example UIs used in the Gate and Runner module are shown in Fig. 15, Fig.16 and Fig.17 to illustrate the concepts. The graphical interactions within CAD environment are not included.

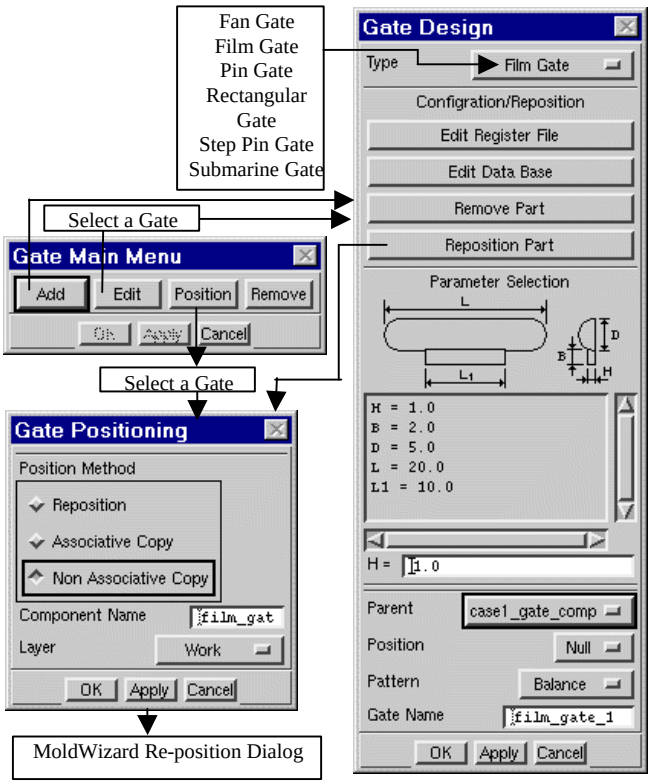
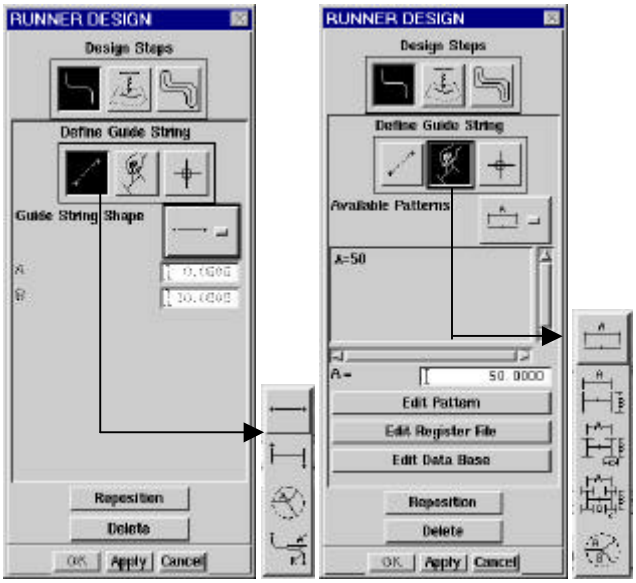


Fig.15 Main UI for gate design module

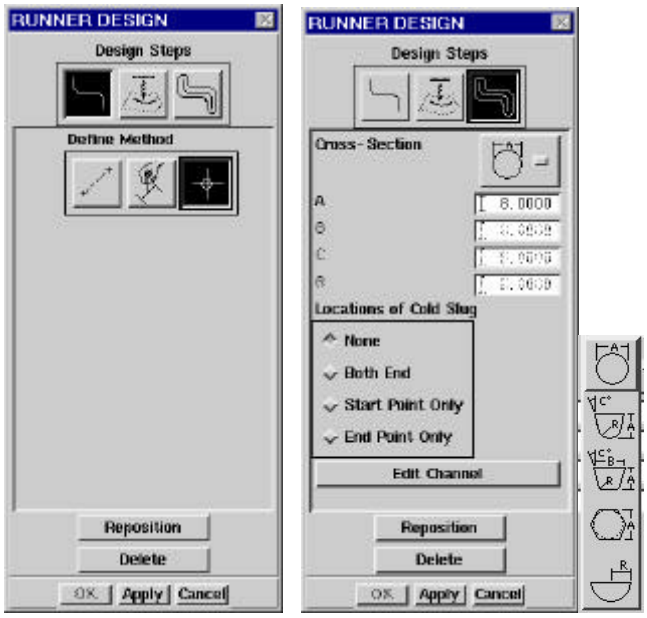


(a) Guiding string shapes (b) Guiding string patterns

Fig.16 Runner creation UIs

VII. Conclusion

In this paper, four modules of MoldWizard™ are introduced. The mold design rules are embedded in the algorithms of these modules. The design ideas are implemented in a generic modular manner. The emphasis is put on the "design procedures" and “built-in” engineering consideration, especially related to the geometrical association to the plastic part model. User preferred sequences are analyzed and streamlined with very flexible data support has enabled effectiveness and efficiency in the software application. These modules have now been integrated as part of the MoldWizard™ version 2.0, and being widely used all over the world.



(a) Select existing curves as guiding string (b) Runner cross-section patterns

Fig.17 Runner creation UIs

Acknowledgment

The authors would like to acknowledge that the work presented in this paper is the result of teamwork by the Computer Aided Product Technology group of Gintic Institute of Manufacturing Technology, Singapore, with the close technological and technical support from UGS Inc, Cypress, USA. The actual implementation of MoldWizard™ may be updated or enhanced hence it can be different from the content of this paper.

References

[1] "Virtual Enterprise Integration: Creating a Sustainable Manufacturing Life Cycle", CASA/SME Tech Trend Report 2000, Web Site: www.sme.org/casa.

[2] G. A. Britton, Y. S. Ma, and S. B. Tor, "Object Technology Development and Unigraphics", Proceedings of Unigraphics User Group 1999 Spring Conference: Managing Design Evolution, New Beach, California, USA.

[3] EDS Inc. "User Guide for Unigraphices v18 (User Functions)".

[4] R. G. W. Pye, *Injection Mould Design*. The Plastic and Rubber Institute, London and New York, 1982.