Knowledge Encapsulation and Application in MoldWizard

Dr. Ma YongSheng

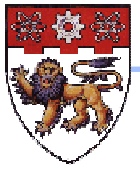
Associate Professor

Nanyang Technological University

Republic of Singapore

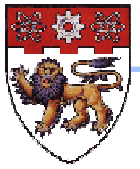
mysma@ntu.edu.sg

Tel: (+65)7905913



Agenda

- Why Knowledge Driven Automation?
- What is MoldWizard™?
- Four Modules of MoldWizard™
  - Cooling Channels
  - Gate and Runner
  - Sub-Insert
  - Electrode
- Discussion on Knowledge Encapsulation and Application
- Conclusion

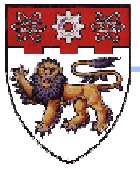


Why Knowledge Driven Automation?

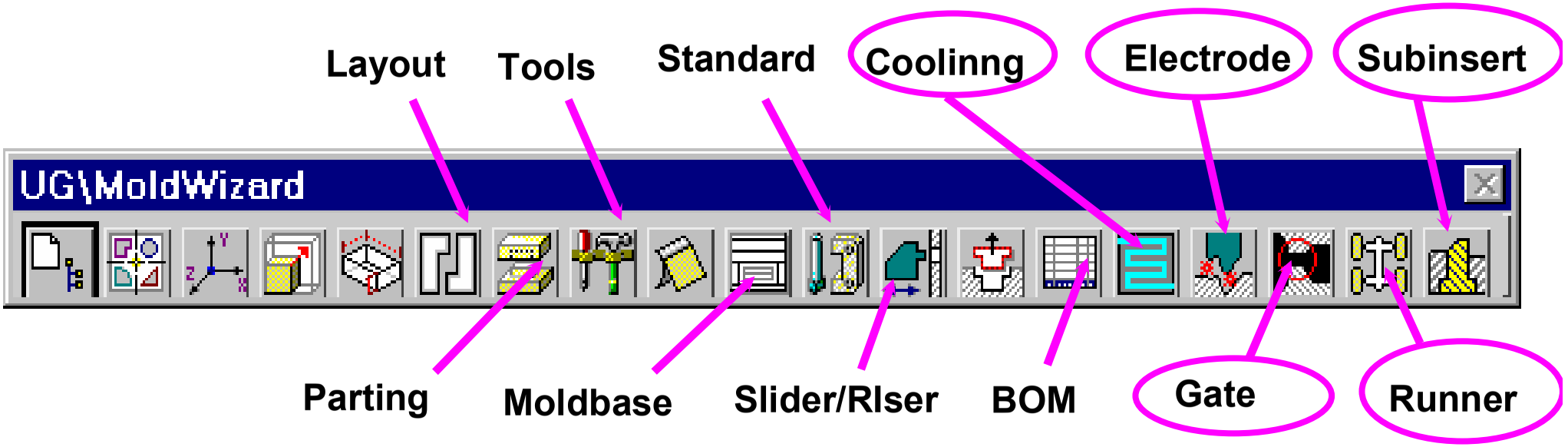
- Parametric feature based solid modeling becomes commodities
- Smart design tools are differentiators
- Easily customizable and automated systems are highly demanded
- The winner has To do The job in shorter time
- Data interoperability is The bottleneck
- Design intent has To be implemented systematically
- Late changes are inevitable

What is MoldWizard™

- A wizard is a set of sequenced UI interfaces to guide the users to complete certain interactions with computer systems.
- Extended wizard by building in engineering process knowledge in the form of CAD/CAM operation sequences and software tools
- A special process based wizard for plastic injection mold design



MoldWizard General Introduction

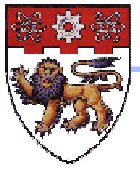


- **12 Main modules**
- **5 main modules developed by Gintic**

Moldwizard V2.0 interface

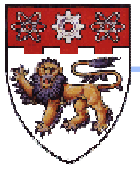
Four Modules of MoldWizard™

- Cooling Channels, Gates and Runners, Sub-Insert and Electrode
- Gintic Institute of Manufacturing Institute was engaged
- Technological/Technical Support from UGS, Cypress
- Four Modules were developed in four months time
- Valuable experiences in methodology and quality
- More challenges ahead



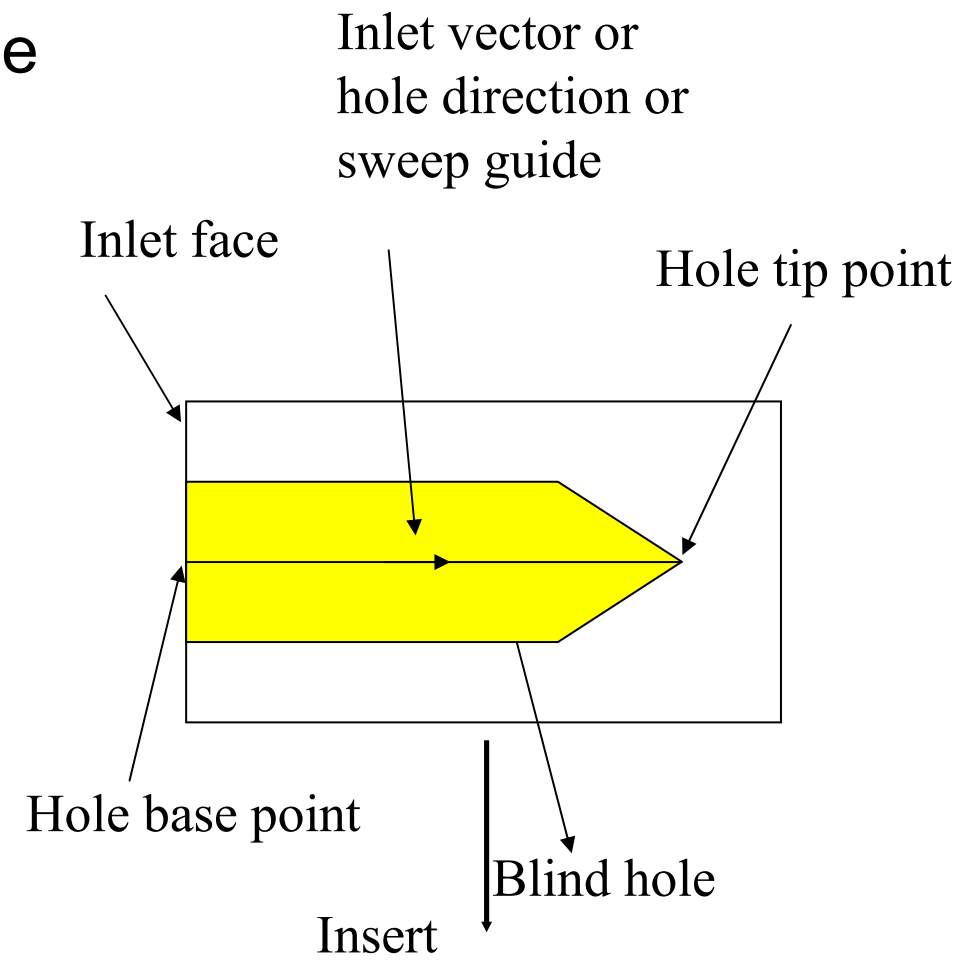
Cooling Channels - Why Not Holes

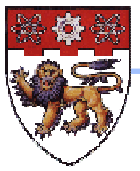
- Time-consuming and repeated manual tasks
- Long design time and low productivity
- To plot cooling circuit drawings without cavity or core block, mold plates, etc
- Repositioning
- Self-Identification (by color, attributes, etc)
- Association among circuit members



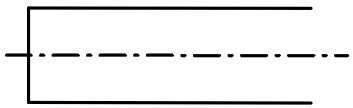
Cooling Channels - Definition

- A simple blind hole

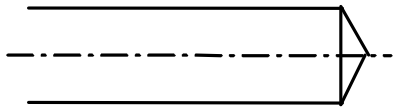




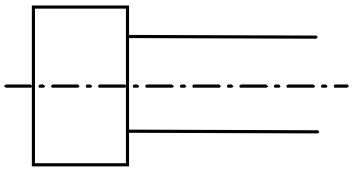
Cooling Channels - Types and Functions



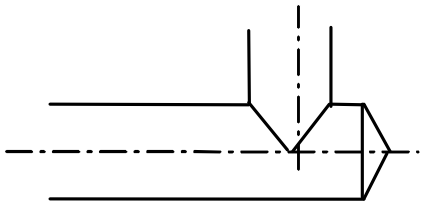
Drill-through end



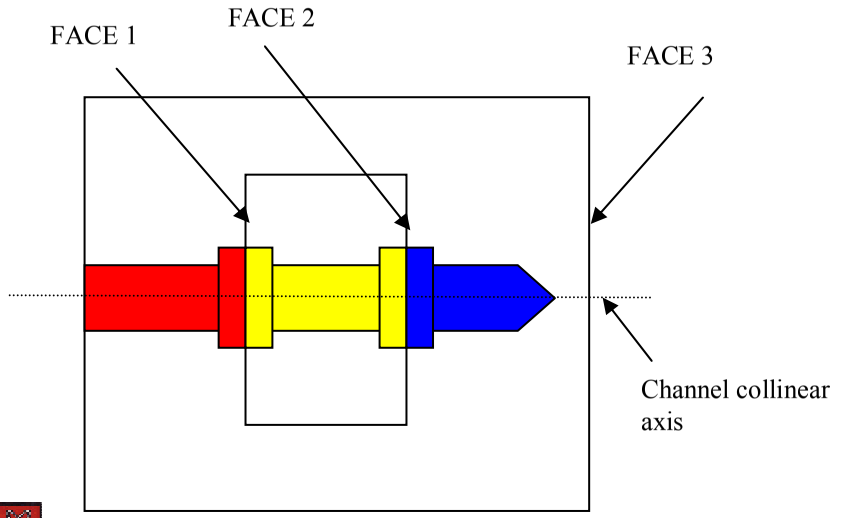
Blind end



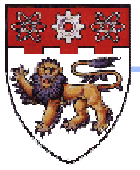
Counter-bored end



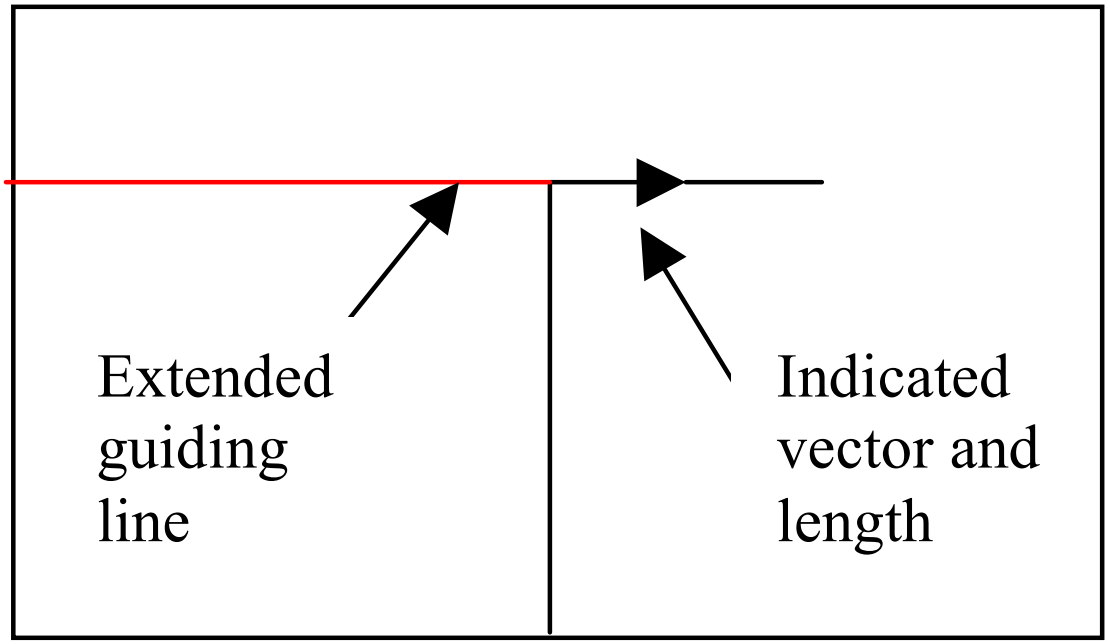
Blind end with extension



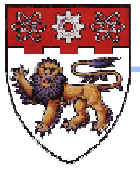
Cooling Channel Module Requirements
Creation of straight cooling channels.
Modification of cooling channel length,
tip angle and diameter.
Creation of U-shaped cooling channel patterns.
Creation of baffle patterns.
Transformation of cooling channels.



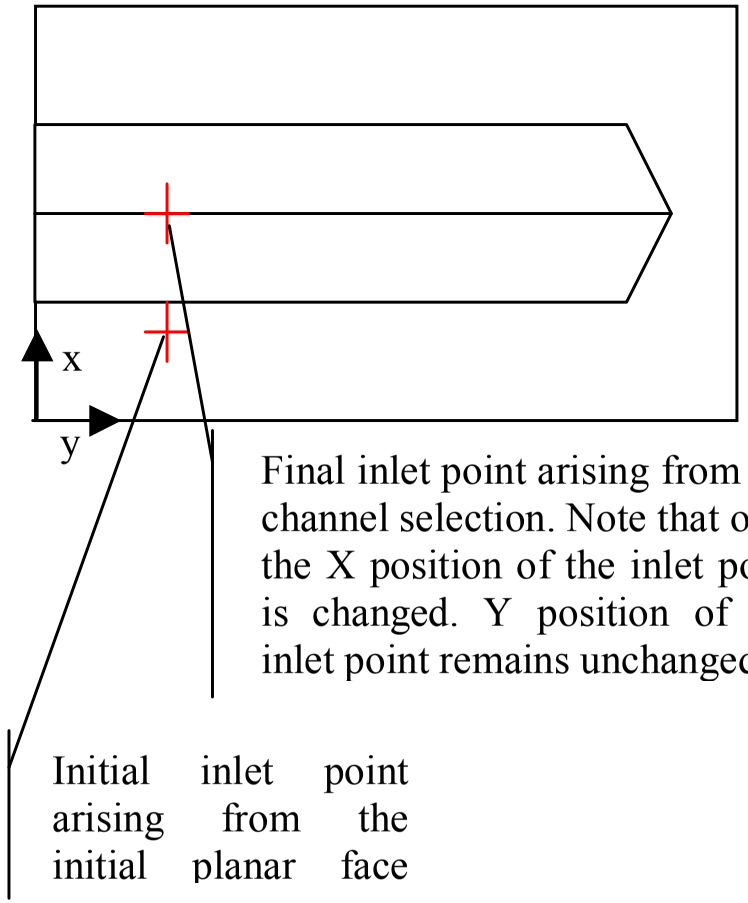
Cooling Channels - Smart Points



Extension of the input line

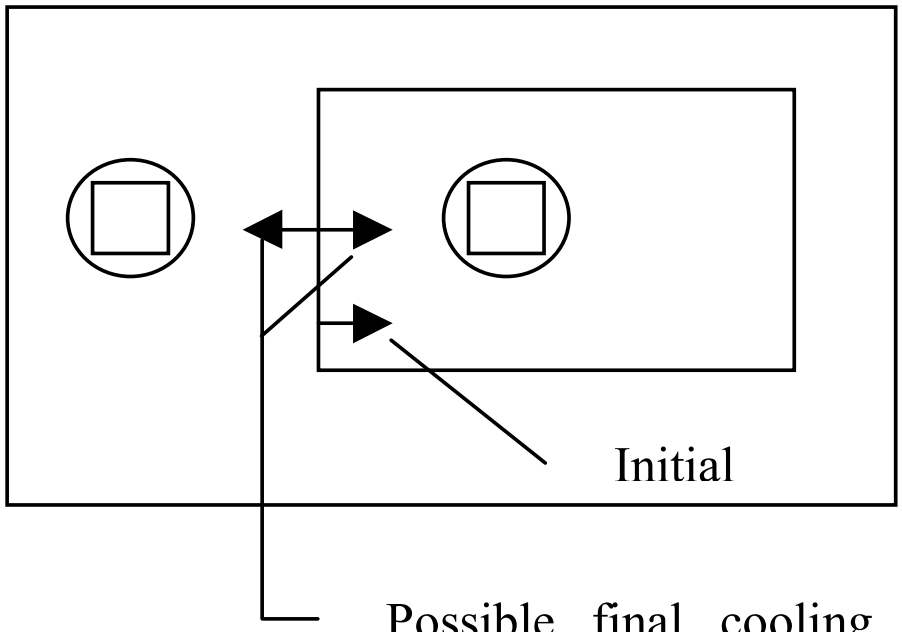


Cooling Channels - Smart Points



Alteration of Inlet Position via Cooling Hole Selection

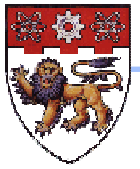
Cooling Channels - Smart Points



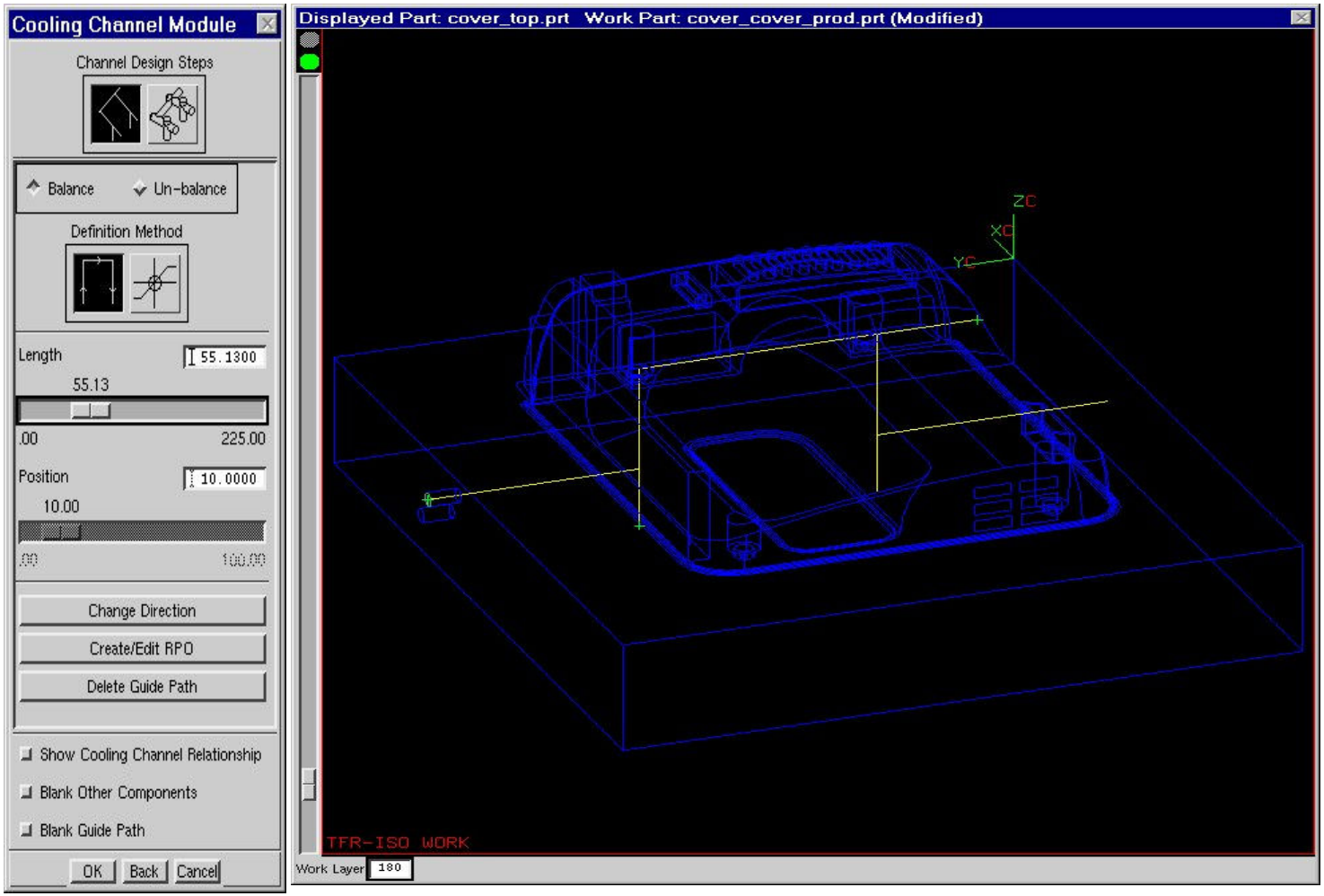
Alteration of Inlet Vector via Cooling Hole Selection

Cooling Channels - Parts and Solids

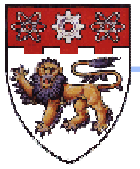
- Create a Cooling Line (CL) part under the top assembly
- Balanced structure --- cooling channels are created under the product part, related waved guide paths and solids are created in CL part
- Unbalanced structure -- when user selects a face in core/cavity, a waved face will be created in the CL part, and smart objects, guide paths and cooling solids will be created in the same part
- Cooling solids are associative to the corresponding penetrating faces



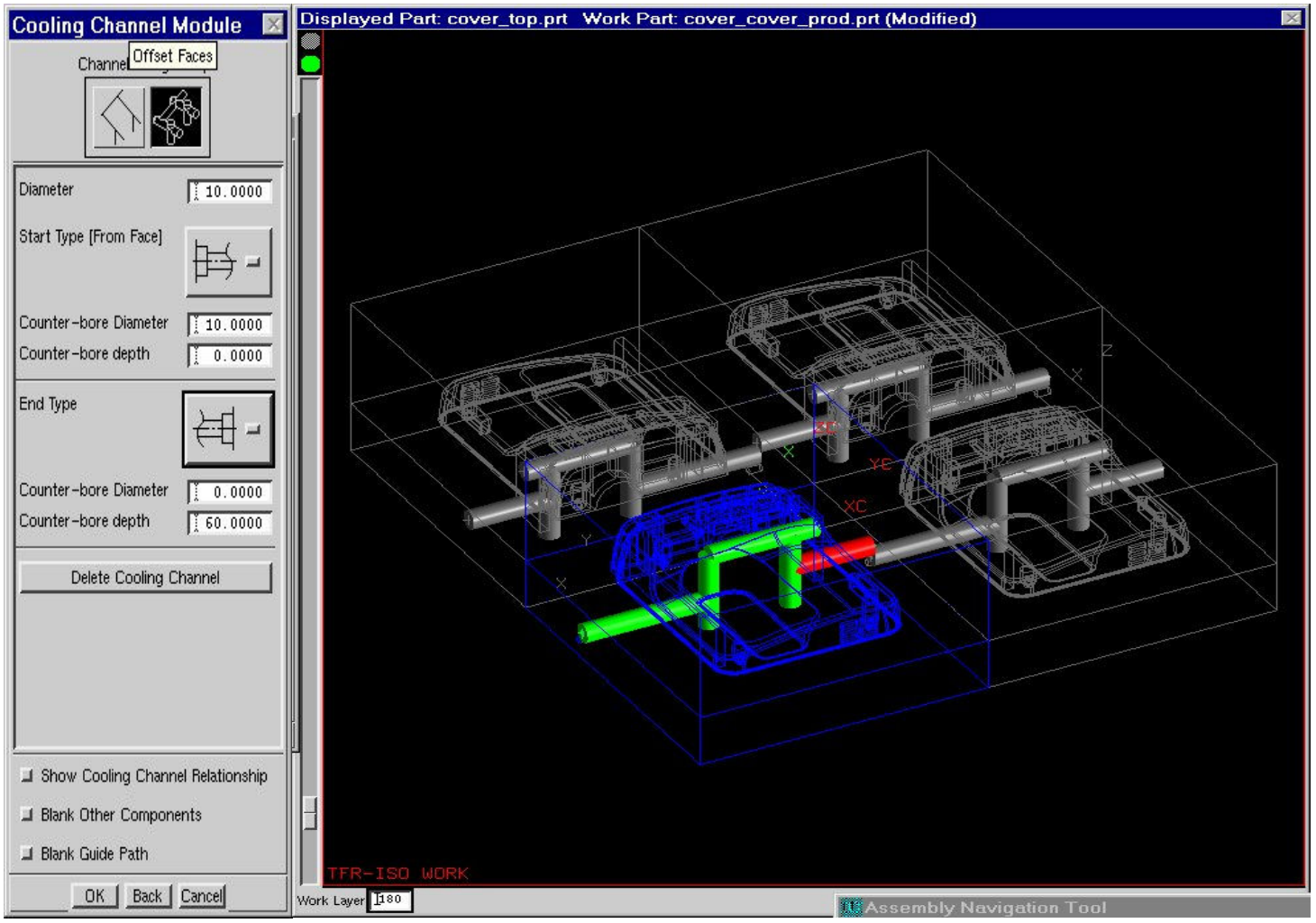
Cooling guide path



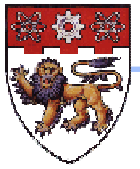
- Guide path of a cooling circuit can be designed continuously
- The lengths and positions of cooling holes can be dynamically edited



Cooling Solids

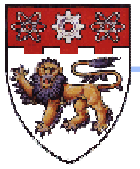


- Parameters of holes can be easily edited
- Types of cooling holes can be modified during this stage
- Guide path can be edited before or after cooling channel creation



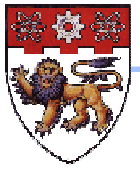
Gates and Runners - Requirements

- Need consistent gate design in the practice
- Parametric and re-usable typical gates
- The creation of the runner system
  - modeling the primary runner, branch runners and cold slug wells
- Substantial number of interactive operations
  - creation of guide strings (curves), cross-sectional curves and a host of other features.
- Multiple cavities



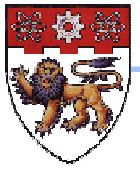
Gates and Runners - Requirements

Requirement Description
A library of commonly-used, parametric gate models.
Modification gate parameters via a user-friendly UI dialog.
Creation of gates for multiple cavities with or without associativity.
Positioning of gates via a user-friendly UI dialog.
Creation of H-shaped, O-shaped or S-shaped guide string patterns for runner creation.
Creation of runner channels and cold slug wells.
Modification of runner channels.



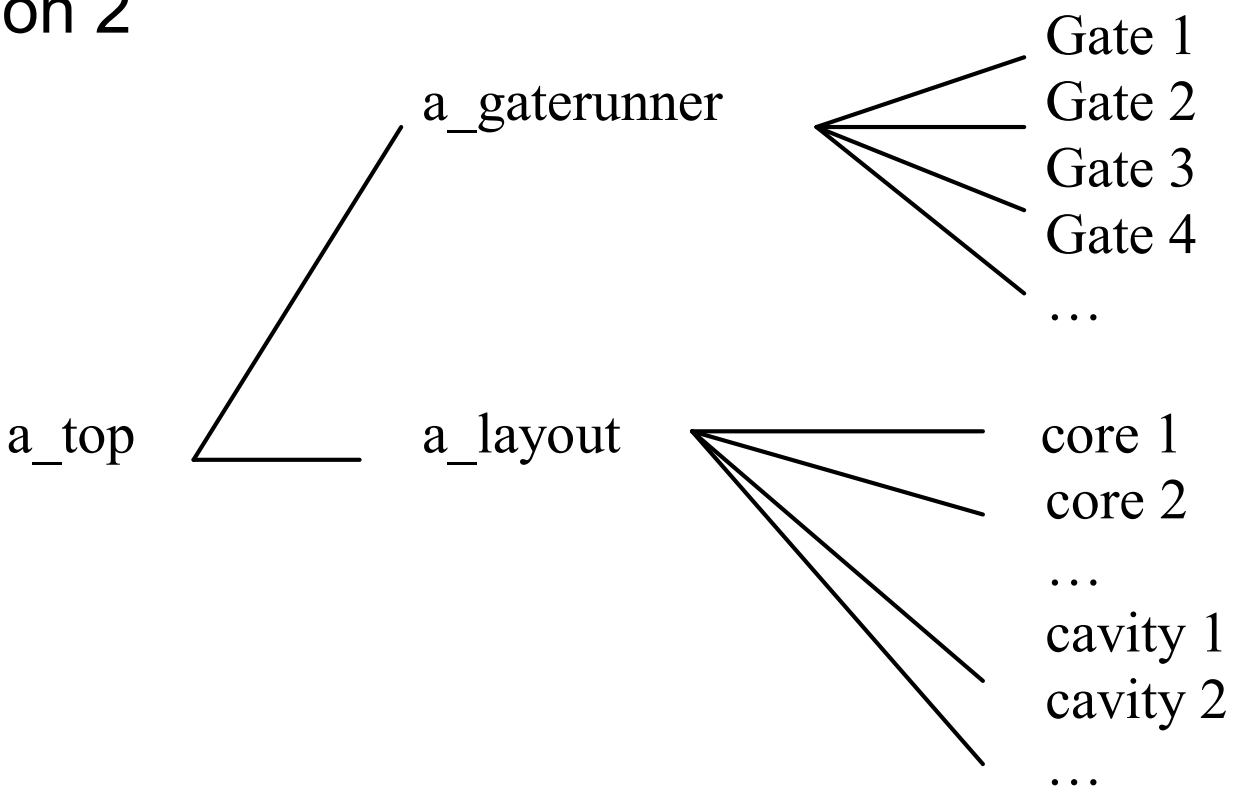
How to Organize Gates - Representation

- Option 1 -- all gate solids and are accommodated in a single gate/runner component under the top assembly.
- 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.
- Simple assembly tree
- Disadvantages:
  - » not able to associate with any smart objects because of UF: reference point is fixed values rather than an object pointer.
  - » Cannot update group gates in one go with native UG functions out of the module
  - » Gates does not follow the layout changes
  - » Gates does not follow each other for re-positioning in native UG

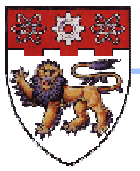


How to Organize Gates - Representation

- Option 2

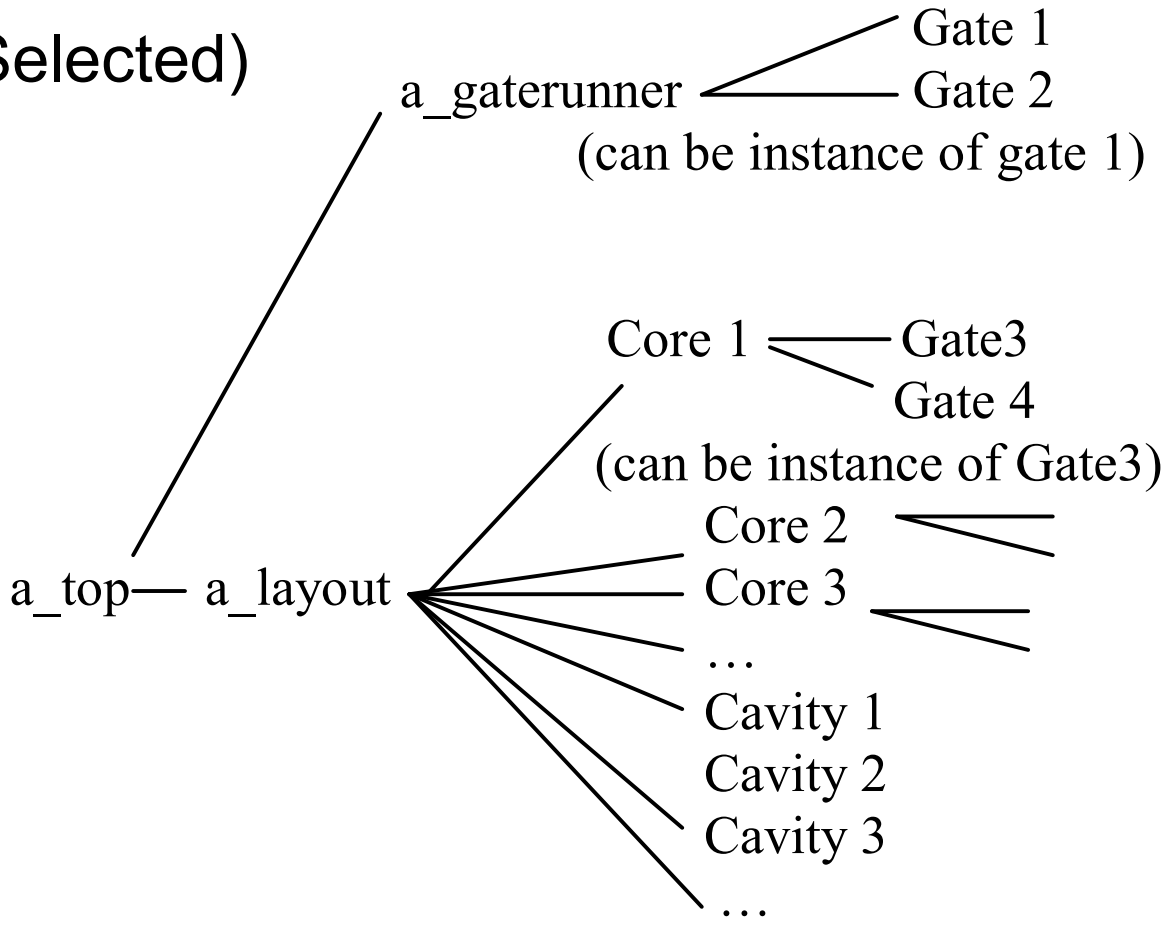


Assembly structure of option 2



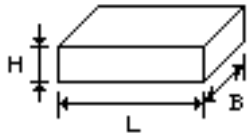
How to Organize Gates - Representation

- Option 3 (Selected)

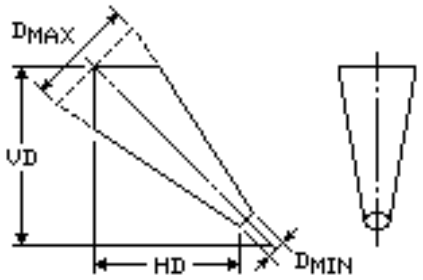


Assembly structure of option 3

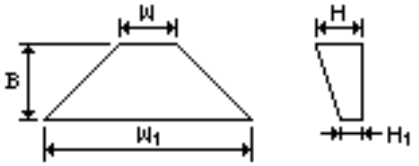
Parametric Gates Types



rectangular gate

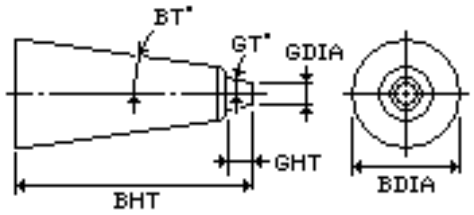


submarine gate

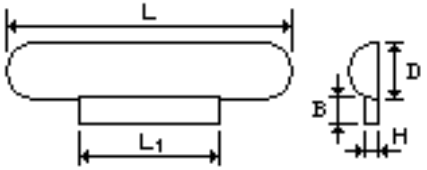


fan gate

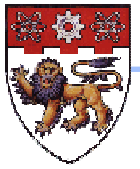
pin gate



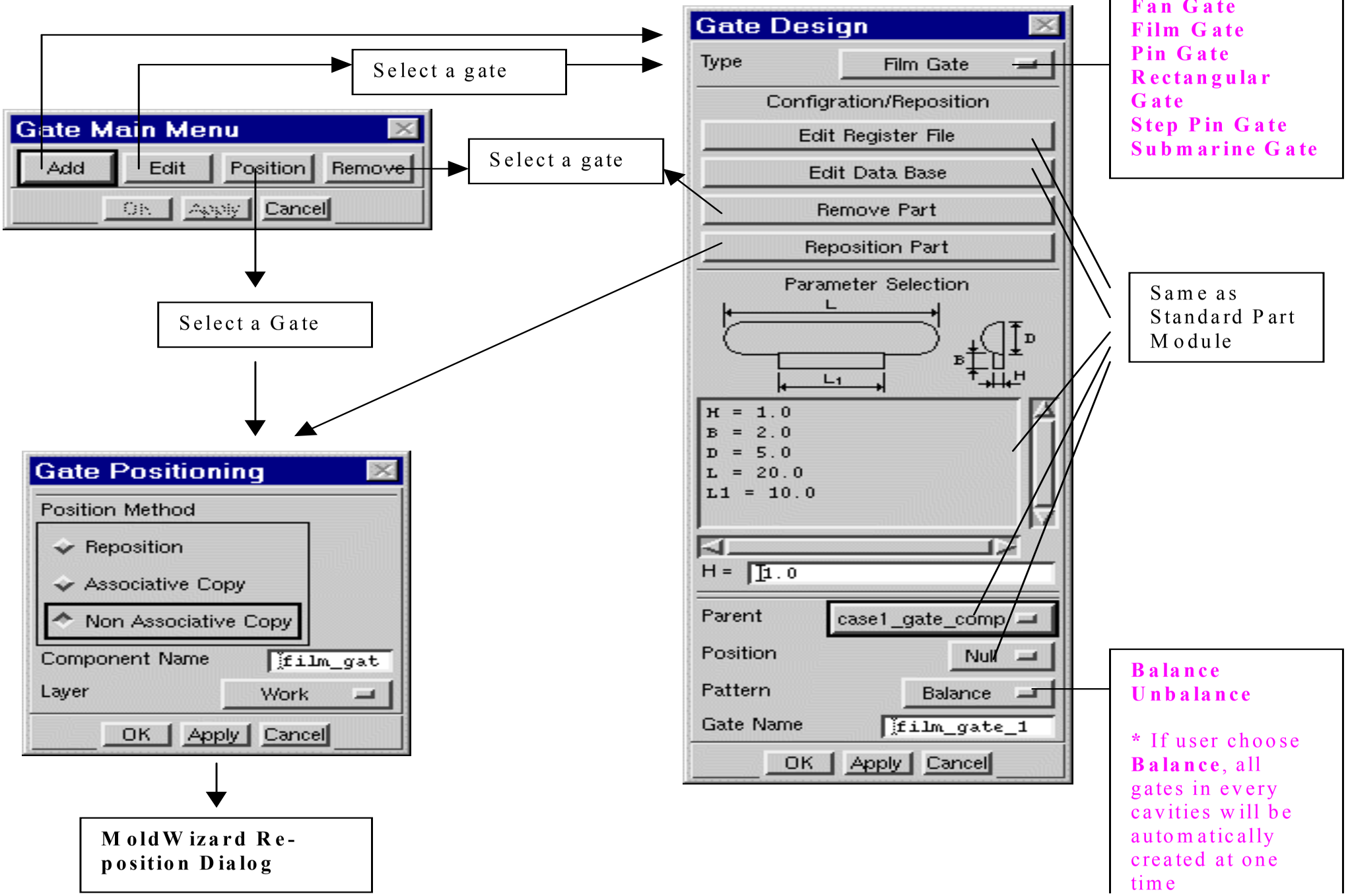
step pin gate



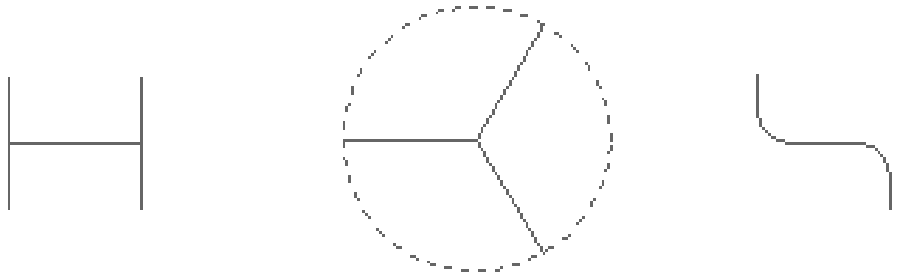
film gate



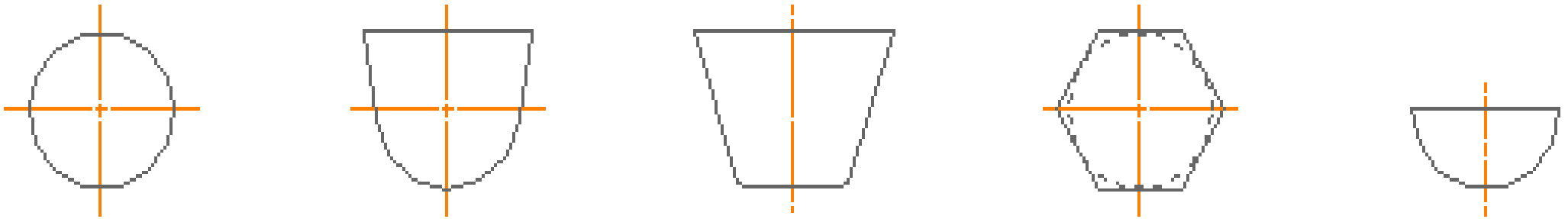
Gates - Interactions



Runners Patterns

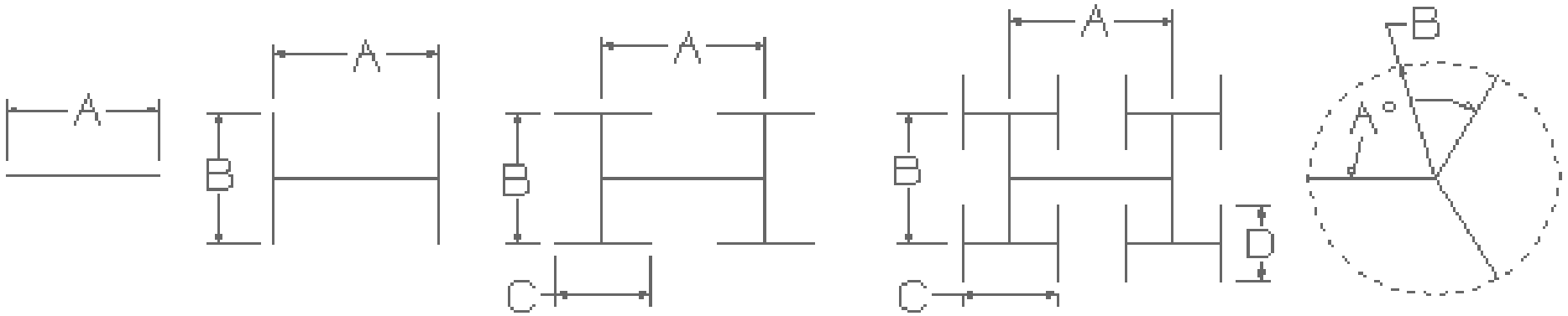


H-shaped O-shaped S-shaped

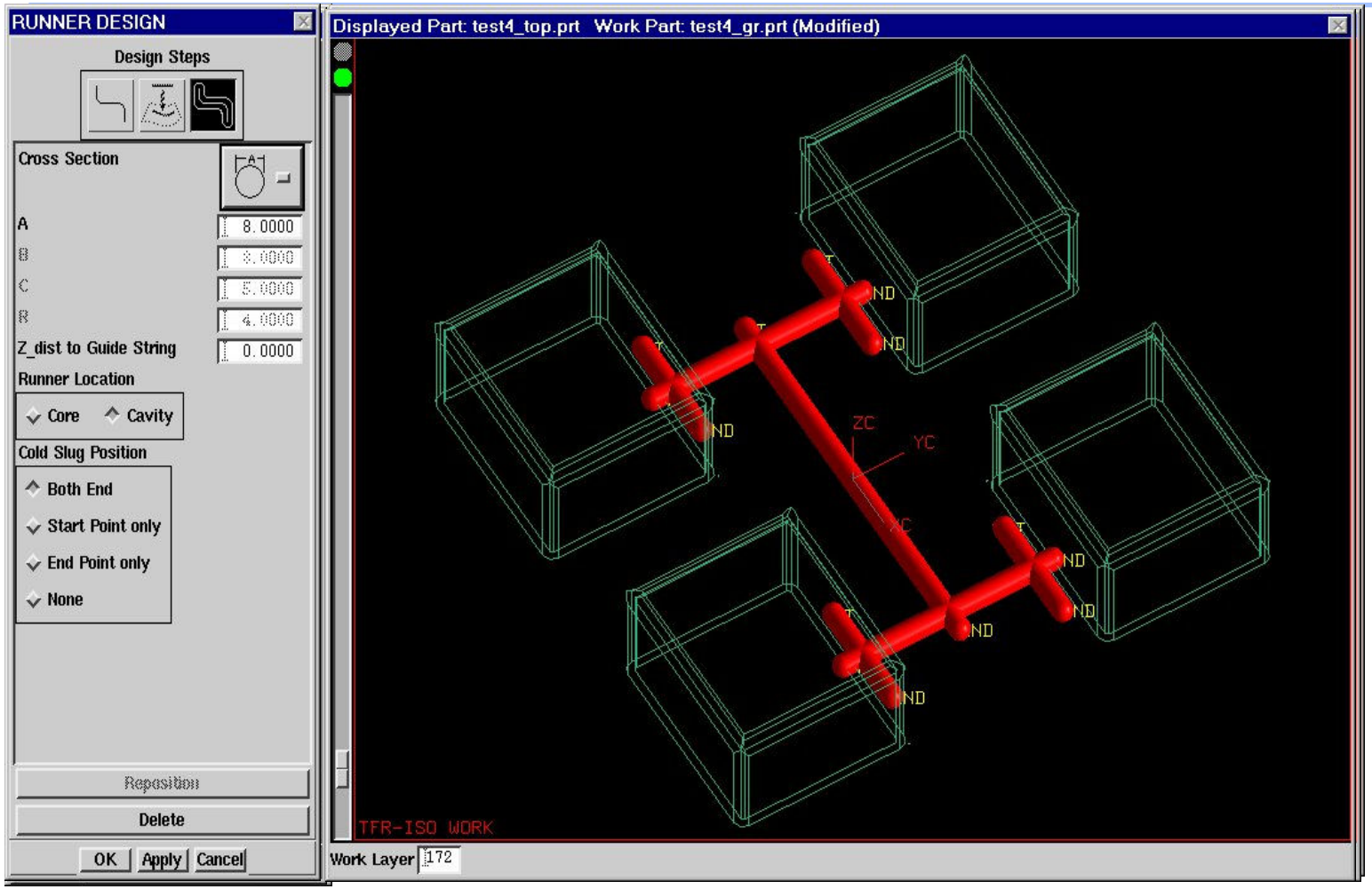


C -- Circular,
P -- Parabolic,
T -- Trapezoidal,
H -- Hexagonal, and
S -- Semi-circular.

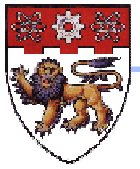
Implemented Runners Patterns



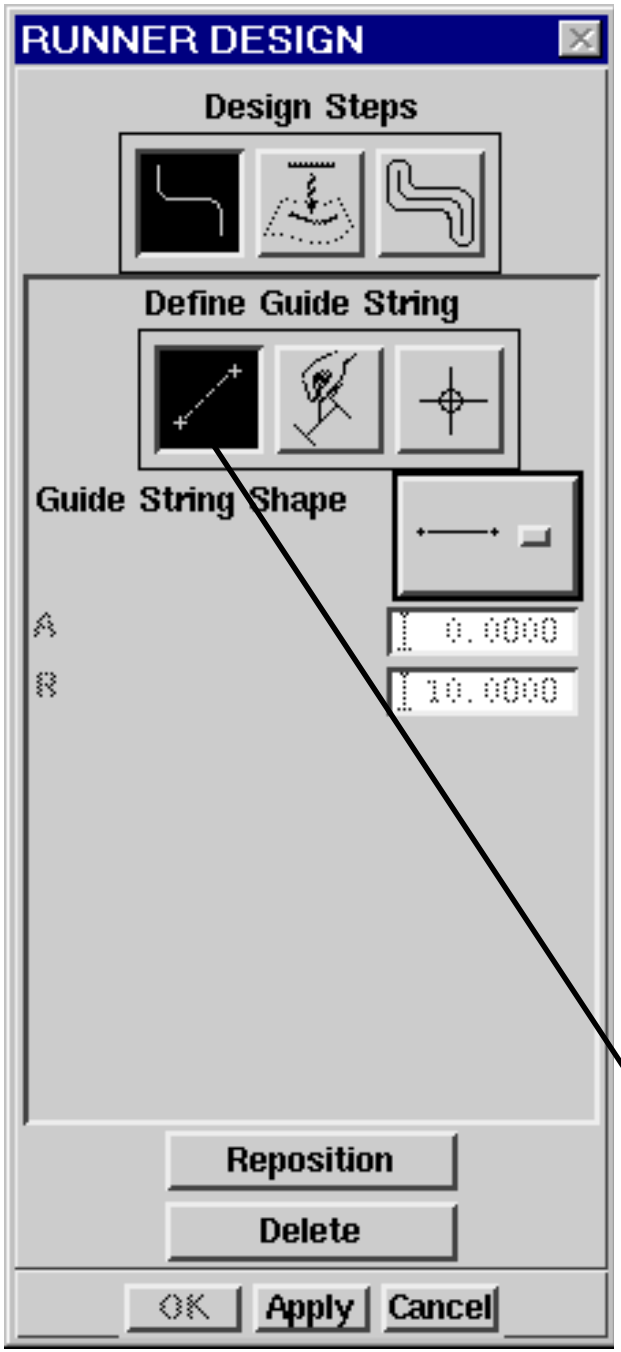
Runner

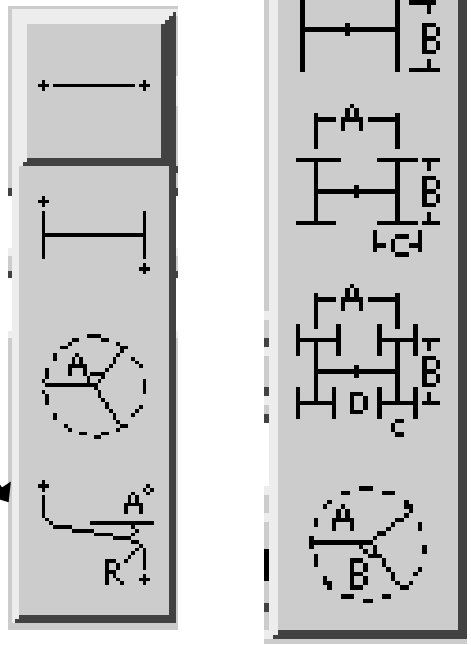


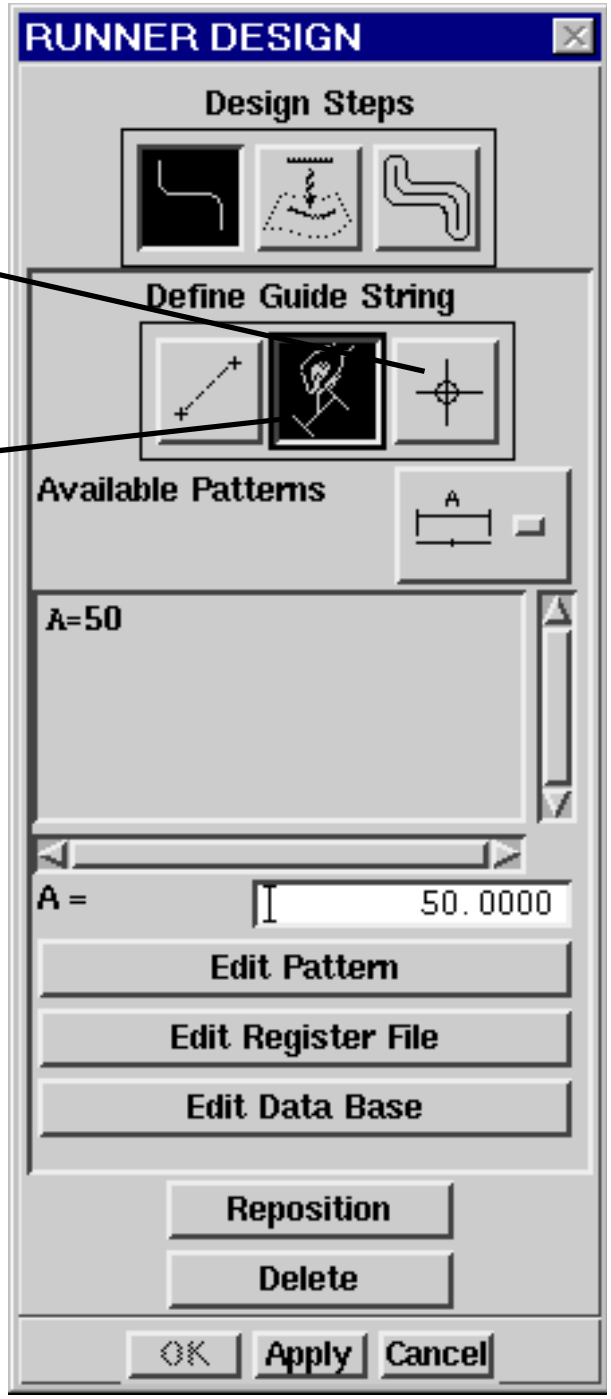
- projection
- 5 types
- Cool well can be auto-attached
- Parameters

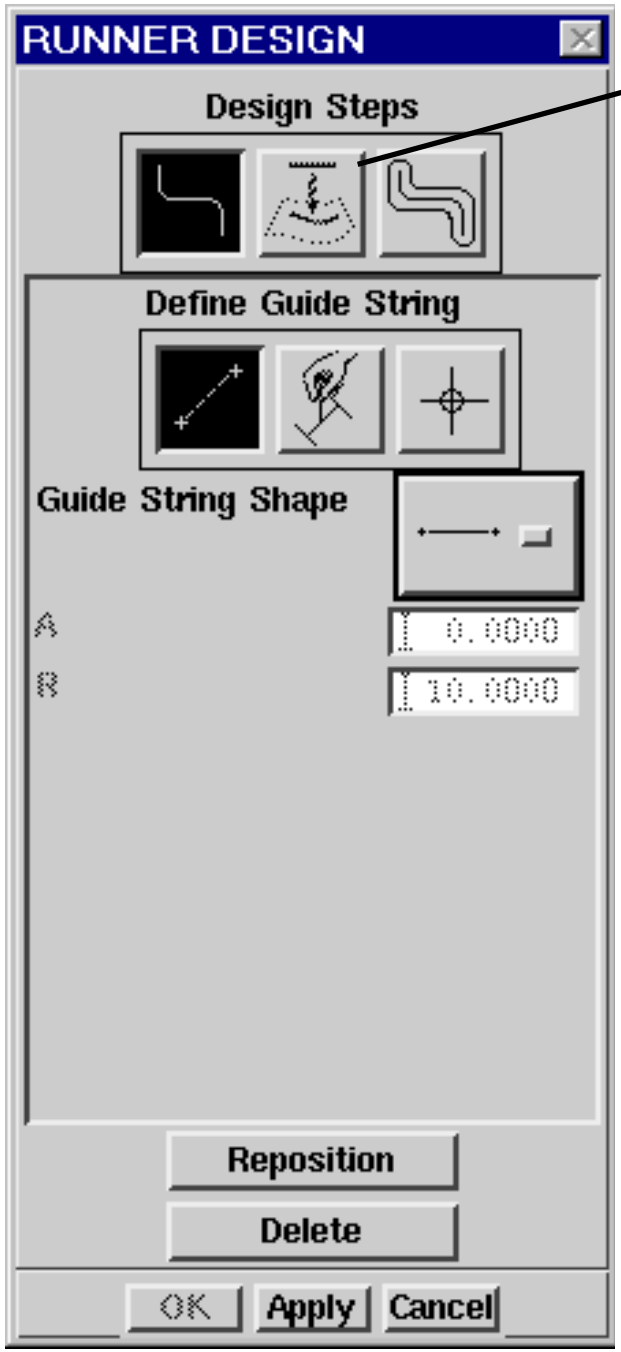


The user can select the existing curves as guide strings.

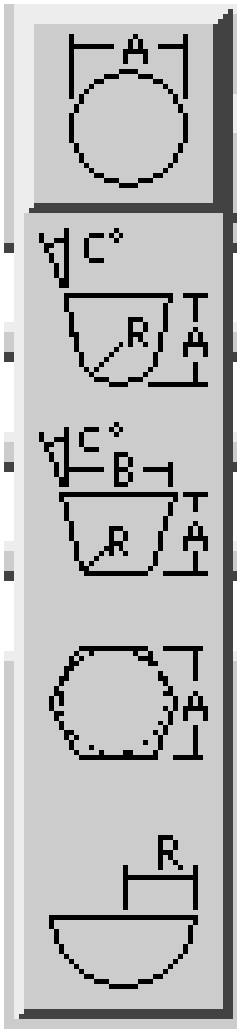


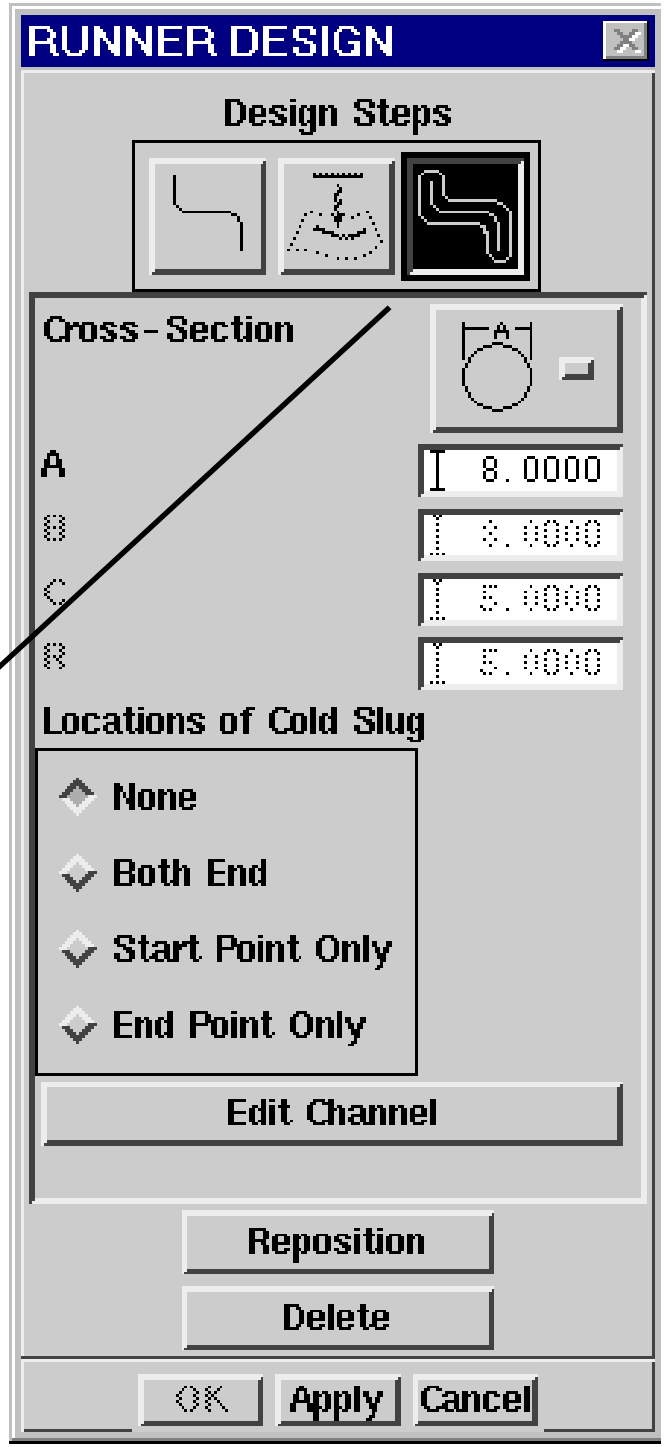






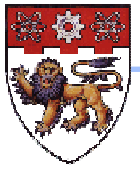
project the planar guide strings onto parting surface when the parting surface is not planar.



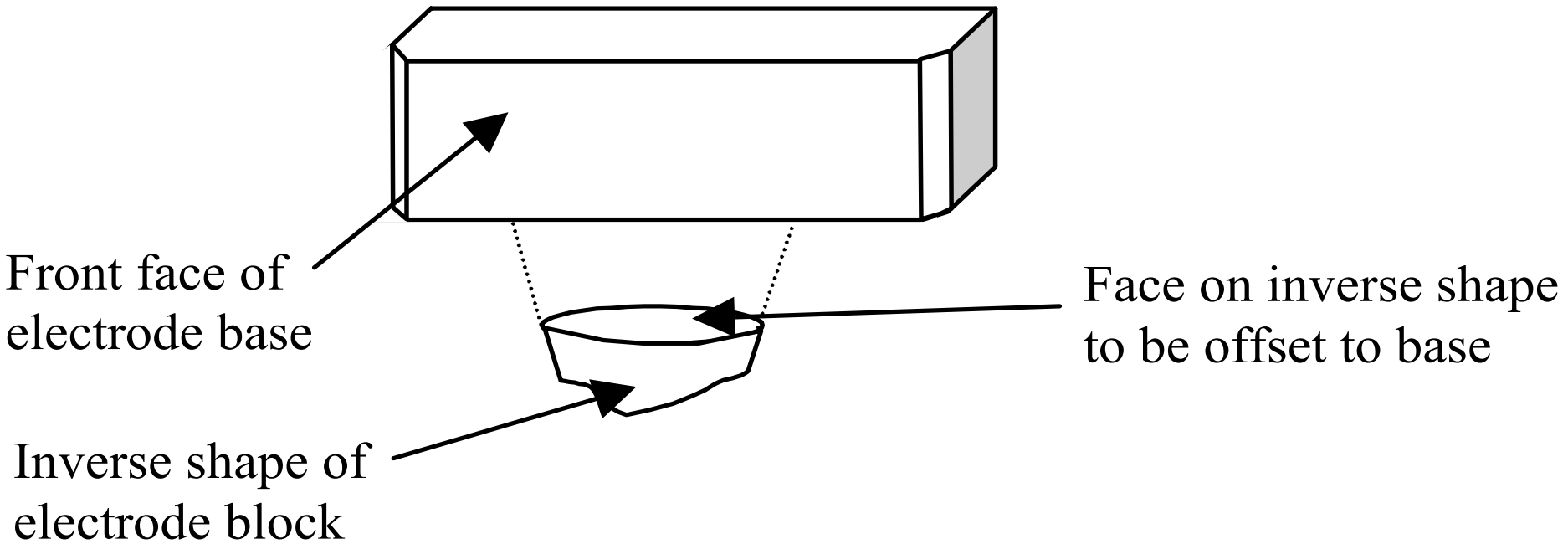


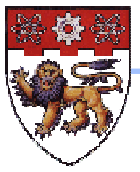
Electrode - Requirements

- 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)
- 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)
- Adding a reference coordinate system to the electrode for machining purposes
- 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.

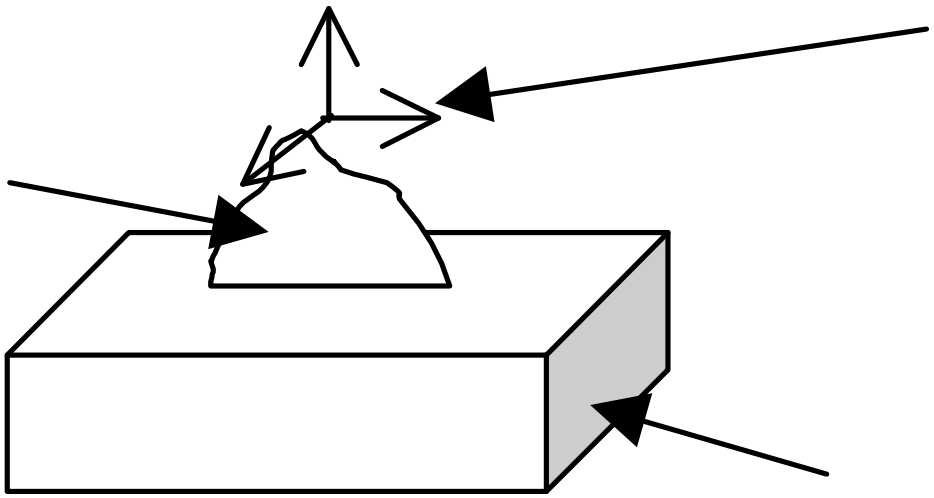


Electrode - Definition

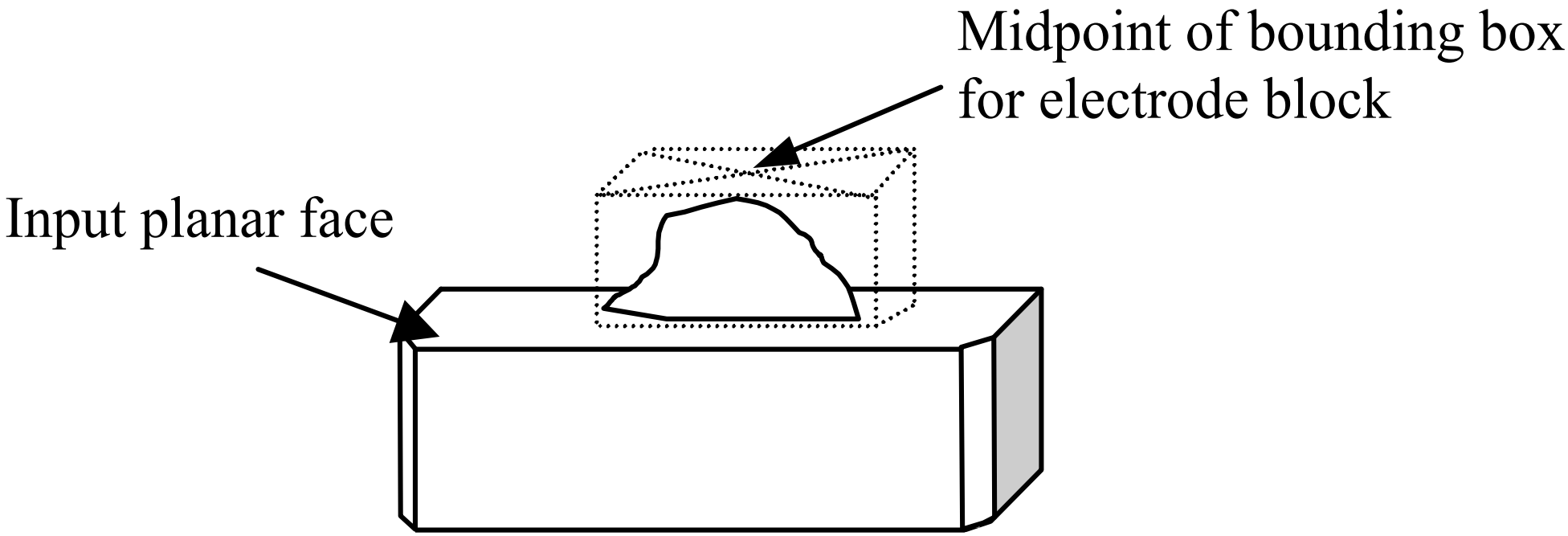




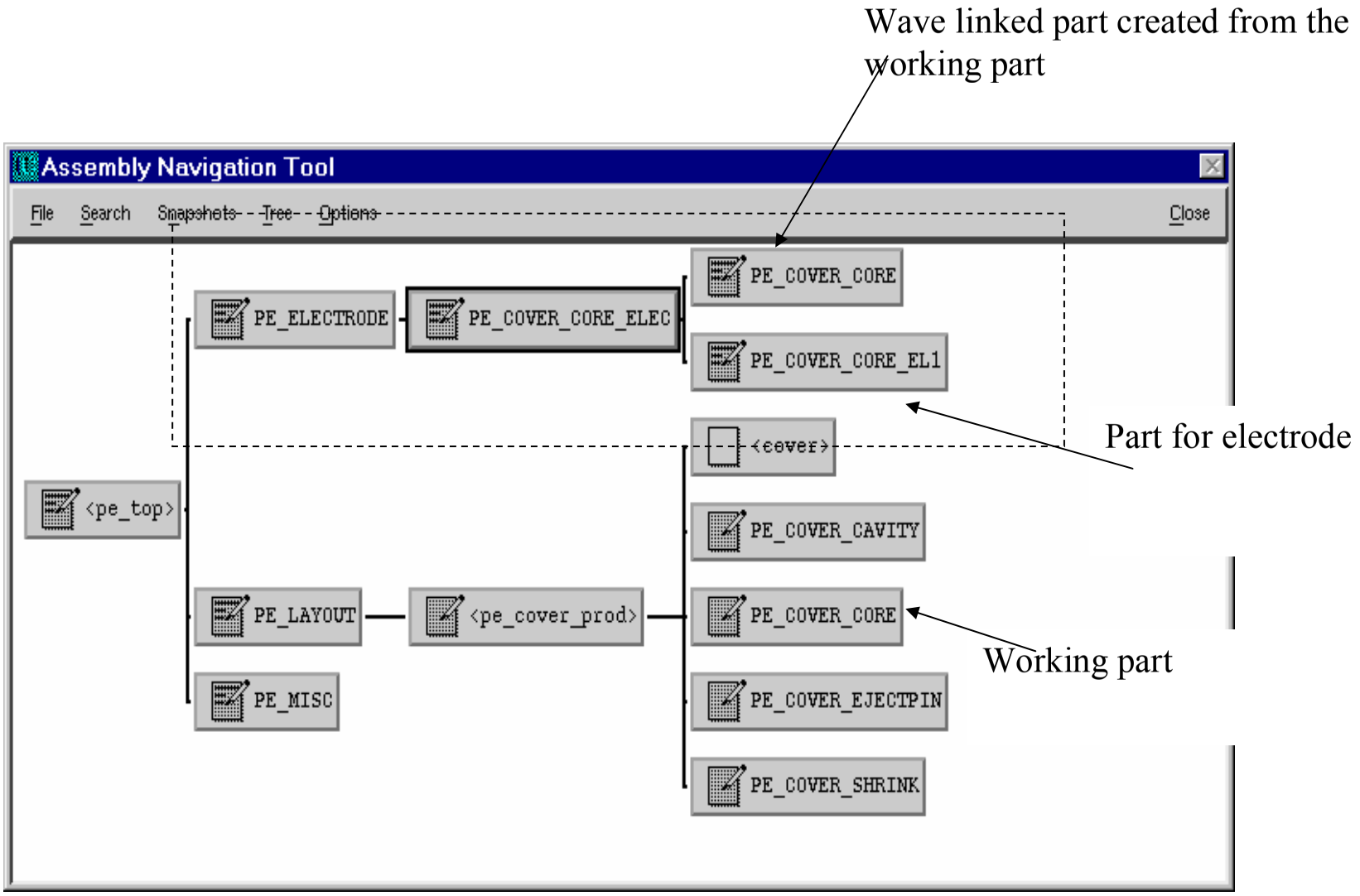
Electrode - Definition



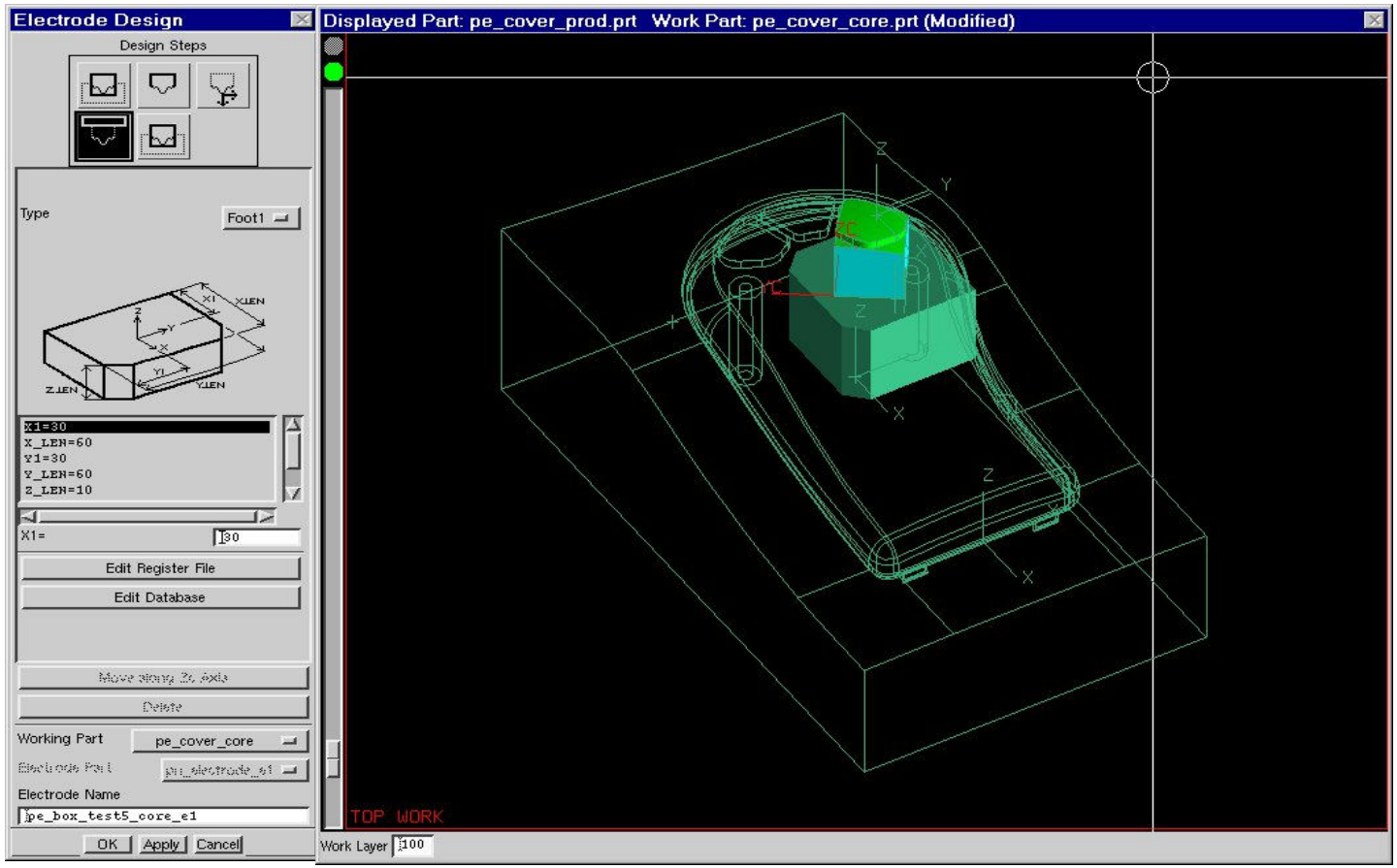
Electrode - Definition



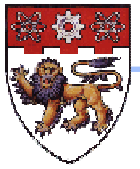
Electrode - Assembly Structure



Electrode(Head/Foot)



- Box can be trimmed by parting face/ sheet/ face
- Box can be created by boolean operation with other solid
- There is an association between foot and head
- Sizes of foot can be easily adjusted

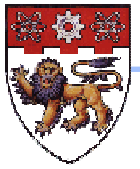


Sub-Insert - Requirements

The process of creating the entire sub-insert can be divided into the following tasks:

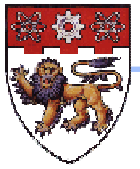
- creation of sub-insert head
- creation of sub-insert body
- creation of positioning and orientation features
- creation of fastening features

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.



Discussion on Knowledge Encapsulation and Application

- What is knowledge in CAD context?
- Format to represent knowledge, e.g. databases, algorithms, sketches, pre-defined geometry, libraries
- Generic algorithms?
- Objects?
- Neutral language and data exchange standard?
- Rules?
- CAD -> KDA->KBE
- KBE->KDA->CAD
- Solutions?



Conclusion

- KDA has great potential
- More research and development is required
- KDA based solutions are highly demanded
- KDA will change the business nature
- KDA globalization
- Knowledge distribution and APS model