Int J Adv Manuf Technol (2000) 16:739–747

The International Journal of
Advanced
Manufacturing
Technology

Automated Assembly Modelling for Plastic Injection Moulds

X. G. Ye, J. Y. H. Fuh and K. S. Lee

Department of Mechanical and Production Engineering, National University of Singapore, Singapore

An injection mould is a mechanical assembly that consists of product-dependent parts and product-independent parts. This paper addresses the two key issues of assembly modelling for injection moulds, namely, representing an injection mould assembly in a computer and determining the position and orientation of a product-independent part in an assembly. A feature-based and object-oriented representation is proposed to represent the hierarchical assembly of injection moulds. This representation requires and permits a designer to think beyond the mere shape of a part and state explicitly what portions of a part are important and why. Thus, it provides an opportunity for designers to design for assembly (DFA). A simplified symbolic geometric approach is also presented to infer the configurations of assembly objects in an assembly according to the mating conditions. Based on the proposed representation and the simplified symbolic geometric approach, automatic assembly modelling is further discussed.

Keywords: Assembly modelling; Feature-based; Injection moulds; Object-oriented

1. Introduction

Injection moulding is the most important process for manufacturing plastic moulded products. The necessary equipment consists of two main elements, the injection moulding machine and the injection mould. The injection moulding machines used today are so-called universal machines, onto which various moulds for plastic parts with different geometries can be mounted, within certain dimension limits, but the injection mould design has to change with plastic products. For different moulding geometries, different mould configurations are usually necessary. The primary task of an injection mould is to shape the molten material into the final shape of the plastic product. This task is fulfilled by the cavity system that consists of core, cavity, inserts, and slider/lifter heads. The geometrical shapes and sizes of a cavity system are determined directly by the plastic moulded product, so all components of a cavity system are called product-dependent parts. (Hereinafter, *product* refers to a plastic moulded product, *part* refers to the component of an injection mould.) Besides the primary task of shaping the product, an injection mould has also to fulfil a number of tasks such as the distribution of melt, cooling the molten material, ejection of the moulded product, transmitting motion, guiding, and aligning the mould halves. The functional parts to fulfil these tasks are usually similar in structure and geometrical shape for different injection moulds. Their structures and geometrical shapes are independent of the plastic moulded products, but their sizes can be changed according to the plastic products. Therefore, it can be concluded that an injection mould is actually a mechanical assembly that consists of product-dependent parts and product-independent parts. Figure 1 shows the assembly structure of an injection mould.

The design of a product-dependent part is based on extracting the geometry from the plastic product. In recent years, CAD/CAM technology has been successfully used to help mould designers to design the product-dependent parts. The

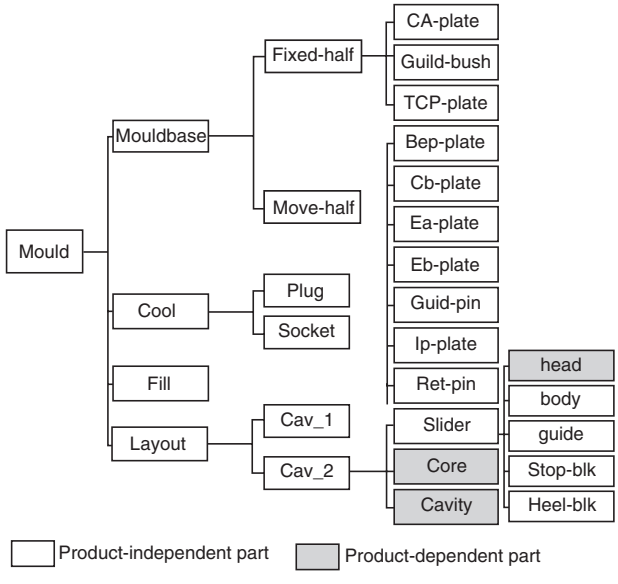


Fig. 1. Assembly structure of an injection mould.

Correspondence and offprint requests to: Dr Jerry Y. H. Fuh, Department of Mechanical and Production Engineering, National University of Singapore (NUS), 10 Kent Ridge Crescent, Singapore 119260. E-mail: mpefuhyh@nus.edu.sg

automatic generation of the geometrical shape for a product-dependent part from the plastic product has also attracted a lot of research interest [1,2]. However, little work has been carried out on the assembly modelling of injection moulds, although it is as important as the design of product-dependent parts. The mould industry is facing the following two difficulties when use a CAD system to design product-independent parts and the whole assembly of an injection mould. First, there are usually around one hundred product-independent parts in a mould set, and these parts are associated with each other with different kinds of constraints. It is time-consuming for the designer to orient and position the components in an assembly. Secondly, while mould designers, most of the time, think on the level of real-world objects, such as screws, plates, and pins, the CAD system uses a totally different level of geometrical objects. As a result, high-level object-oriented ideas have to be translated to low-level CAD entities such as lines, surfaces, or solids. Therefore, it is necessary to develop an automatic assembly modelling system for injection moulds to solve these two problems. In this paper, we address the following two key issues for automatic assembly modelling: representing a product-independent part and a mould assembly in a computer; and determining the position and orientation of a component part in an assembly.

This paper gives a brief review of related research in assembly modelling, and presents an integrated representation for the injection mould assembly. A simplified geometric symbolic method is proposed to determine the position and orientation of a part in the mould assembly. An example of automatic assembly modelling of an injection mould is illustrated.

2. Related Research

Assembly modelling has been the subject of research in diverse fields, such as, kinematics, AI, and geometric modelling. Libardi et al. [3] compiled a research review of assembly modelling. They reported that many researchers had used graph structures to model assembly topology. In this graph scheme, the components are represented by nodes, and transformation matrices are attached to arcs. However, the transformation matrices are not coupled together, which seriously affects the transformation procedure, i.e. if a subassembly is moved, all its constituent parts do not move correspondingly. Lee and Gossard [4] developed a system that supported a hierarchical assembly data structure containing more basic information about assemblies such as "mating feature" between the components. The transformation matrices are derived automatically from the associations of virtual links, but this hierarchical topology model represents only "part-of" relations effectively.

Automatically inferring the configuration of components in an assembly means that designers can avoid specifying the transformation matrices directly. Moreover, the position of a component will change whenever the size and position of its reference component are modified. There exist three techniques to infer the position and orientation of a component in the assembly: iterative numerical technique, symbolic algebraic technique, and symbolic geometric technique. Lee and Gossard [5] proposed an iterative numerical technique to compute the location and orientation of each component from the spatial relationships. Their method consists of three steps: generation of the constraint equations, reducing the number of equations, and solving the equations. There are 16 equations for "*against*" condition, 18 equations for "*fit*" condition, 6 property equations for each matrix, and 2 additional equations for a rotational part. Usually the number of equations exceeds the number of variables, so a method must be devised to remove the redundant equations. The Newton–Raphson iteration algorithm is used to solve the equations. This technique has two disadvantages: first, the solution is heavily dependent on the initial solution; secondly, the iterative numerical technique cannot distinguish between different roots in the solution space. Therefore, it is possible, in a purely spatial relationship problem, that a mathematically valid, but physically unfeasible, solution can be obtained.

Ambler and Popplestone [6] suggested a method of computing the required rotation and translation for each component to satisfy the spatial relationships between the components in an assembly. Six variables (three translations and three rotations) for each component are solved to be consistent with the spatial relationships. This method requires a vast amount of programming and computation to rewrite related equations in a solvable format. Also, it does not guarantee a solution every time, especially when the equation cannot be rewritten in solvable forms.

Kramer [7] developed a symbolic geometric approach for determining the positions and orientations of rigid bodies that satisfy a set of geometric constraints. Reasoning about the geometric bodies is performed symbolically by generating a sequence of actions to satisfy each constraint incrementally, which results in the reduction of the object's available degrees of freedom (DOF). The fundamental reference entity used by Kramer is called a "marker", that is a point and two orthogonal axes. Seven constraints (coincident, in-line, in-plane, parallel_z, offset_z, offset_x and helical) between markers are defined. For a problem involving a single object and constraints between markers on that body, and markers which have invariant attributes, action analysis [7] is used to obtain a solution. Action analysis decides the final configuration of a geometric object, step by step. At each step in solving the object configuration, degrees of freedom analysis decides what action will satisfy one of the body's as yet unsatisfied constraints, given the available degrees of freedom. It then calculates how that action further reduces the body's degrees of freedom. At the end of each step, one appropriate action is added to the metaphorical assembly plan. According to Shah and Rogers [8], Kramer's work represents the most significant development for assembly modelling. This symbolic geometric approach can locate all solutions to constraint conditions, and is computationally attractive compared to an iterative technique, but to implement this method, a large amount of programming is required.

Although many researchers have been actively involved in assembly modelling, little literature has been reported on feature based assembly modelling for injection mould design. Kruth et al. [9] developed a design support system for an injection mould. Their system supported the assembly design for injection moulds through high-level functional mould objects (components and features). Because their system was

based on AutoCAD, it could only accommodate wire-frame and simple solid models.

3. Representation of Injection Mould Assemblies

The two key issues of automated assembly modelling for injection moulds are, representing a mould assembly in computers, and determining the position and orientation of a product-independent part in the assembly. In this section, we present an object-oriented and feature-based representation for assemblies of injection moulds.

The representation of assemblies in a computer involves structural and spatial relationships between individual parts. Such a representation must support the construction of an assembly from all the given parts, changes in the relative positioning of parts, and manipulation of the assembly as a whole. Moreover, the representations of assemblies must meet the following requirements from designers:

1. It should be possible to have high-level objects ready to use while mould designers think on the level of real-world objects.
2. The representation of assemblies should encapsulate operational functions to automate routine processes such as pocketing and interference checks.

To meet these requirements, a feature-based and object-oriented hierarchical model is proposed to represent injection moulds. An assembly may be divided into subassemblies, which in turn consists of subassemblies and/or individual components. Thus, a hierarchical model is most appropriate for representing the structural relations between components. A hierarchy implies a definite assembly sequence. In addition, a hierarchical model can provide an explicit representation of the dependency of the position of one part on another.

Feature-based design [10] allows designers to work at a somewhat higher level of abstraction than that possible with the direct use of solid modellers. Geometric features are instanced, sized, and located quickly by the user by specifying a minimum set of parameters, while the feature modeller works out the details. Also, it is easy to make design changes because of the associativities between geometric entities maintained in the data structure of feature modellers. Without features, designers have to be concerned with all the details of geometric construction procedures required by solid modellers, and design changes have to be strictly specified for every entity affected by the change. Moreover, the feature-based representation will provide high-level assembly objects for designers to use. For example, while mould designers think on the level of a real-world object, e.g. a counterbore hole, a feature object of a counterbore hole will be ready in the computer for use.

Object-oriented modelling [11,12] is a new way of thinking about problems using models organised around real-world concepts. The fundamental entity is the object, which combines both data structures and behaviour in a single entity. Object-oriented models are useful for understanding problems and designing programs and databases. In addition, the object-oriented representation of assemblies makes it easy for a "child" object to inherit information from its "parent".

Figure 2 shows the feature-based and object-oriented hierarchical representation of an injection mould. The representation is a hierarchical structure at multiple levels of abstraction, from low-level geometric entities (form feature) to high-level subassemblies. The items enclosed in the boxes represent "assembly objects" (SUB_As, PARTs and FFs); the solid lines represent "part-of" relation; and the dashed lines represent other relationships. Subassembly (SUB_A) consists of parts (PARTs). A part can be thought of as an "assembly" of form features (FFs). The representation combines the strengths of a feature-based geometric model with those of object-oriented models. It not only contains the "part-of" relations between the parent object and the child object, but also includes a richer set of structural relations and a group of operational functions for assembly objects. In Section 3.1, there is further discussion on the definition of an assembly object, and detailed relations between assembly objects are presented in Section 3.2.

3.1 Definition of Assembly Objects

In our work, an assembly object, **O**, is defined as a unique, identifiable entity in the following form:

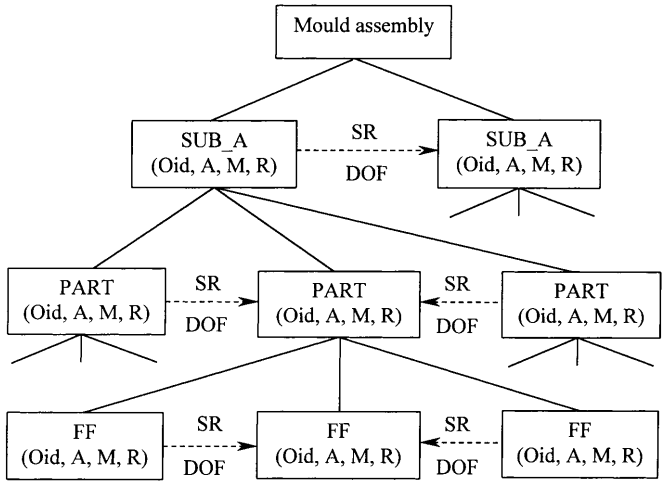
$$\mathbf{O} = (\mathbf{Oid}, \mathbf{A}, \mathbf{M}, \mathbf{R}) \tag{1}$$

Where:

Oid is a unique identifier of an assembly object (**O**).

A is a set of three-tuples, (**t**, **a**, **v**). Each **a** is called an attribute of **O**, associated with each attribute is a *type*, **t**, and a *value*, **v**.

M is a set of tuples, (**m**, $\mathbf{tc}_1$, $\mathbf{tc}_2$, …, $\mathbf{tc}_n$, **tc**). Each element of **M** is a *function* that uniquely identifies a method. The symbol **m** represents a method name; and methods define operations on objects. The symbol $\mathbf{tc}_i$ (*i*



FF: Form feature; **SR**: Spatial relations;
DOF: Degrees of freedom; (Oid, A, M, R): Object definition.

Fig. 2. Feature-based, object-oriented hierarchical representation.

= 1, 2, …, *n*) specifies the argument type and **tc** specifies the returned value type.

R is a set of relationships among **O** and other assembly objects. There are six types of basic relationships between assembly objects, i.e. **Part-of**, **SR**, **SC**, **DOF**, **Lts**, and **Fit**.

Table 1 shows an assembly object of injection moulds, e.g. ejector. The ejector in Table 1 is formally specified as:

(ejector-pin_1, {(string, purpose, 'ejecting moulding'), (string, material, '**nitride steel**'), (string, catalog_no, 'THX')},
{(check_interference(), boolean), (pocket_plate(), boolean)},
{(part-of ejection_sys), (SR Align EB_plate), (DOF Tx, Ty)}).

In this example, *purpose*, *material* and *catalog_no* are attributes with a data type of *string*; *check_interference* and *pocket_plate* are member functions; and *Part-of*, *SR* and *DOF* are relationships.

3.2 Assembly Relationships

There are six types of basic relationships between assembly objects, **Part-of**, **SR**, **SC**, **DOF**, **Lts**, and **Fit**.

Part-of	An assembly object belongs to its ancestor object.
SR	Spatial relations: explicitly specify the positions and orientations of assembly objects in an assembly. For a component part, its spatial relationship is derived from spatial constraints (SC).
SC	Spatial constraints: implicitly locate a component part with respect to the other parts.
DOF	Degrees of freedom: are allowable translational/ rotational directions of motion after assembly, with or without limits.
Lts	Motion limits: because of obstructions/interferences, the **DOF** may have unilateral or bilateral limits.
Fit	Size constraint: is applied to dimensions, in order to maintain a given class of fit.

Table 1. Definition of an assembly object-ejector.

Object **Oid** ejector-pin_1 Instance-of	Ejector_pin		Derived from ejector class
A	Purpose "ejecting moulding"		Type string
	Material "nitrided steel"		Type string
	Catalog_no "THX"		Type string
M	Check_interference (cool_obj)		Check interference between ejectors and cooling lines
	Pocket_plate()		Make a hole on plate to accommodate ejector pins
R	Part-of	ejector_sys	
	SR	align with EB plate	
	DOF	Tx, Ty	

Among all the elements of an assembly object, the relationships are most important for assembly design. The relationships between assembly objects will not only determine the position of objects in an assembly, but also maintain the associativities between assembly objects. In the following sub-sections, we will illustrate the relationships at the same assembly level with the help of examples.

3.2.1 Relationships Between Form Features

Mould design, in essence, is a mental process; mould designers most of the time think on the level of real-world objects such as plates, screws, grooves, chamfers, and counter-bore holes. Therefore, it is necessary to build the geometric models of all product-independent parts from form features. The mould designer can easily change the size and shape of a part, because of the relations between form features maintained in the part representation. Figure 3(*a*) shows a plate with a counter-bore hole. This part is defined by two form features, i.e. a block and a counter-bore hole. The counter-bore hole (FF_2) is placed with reference to the block feature FF_1, using their local coordinates $\mathbf{F}_2$ and $\mathbf{F}_1$, respectively. Equations (2)–(5) show the spatial relationships between the counter-bore hole (FF_2) and the block feature (FF_1). For form features, there is no spatial constraint between them, so the spatial relationships are specified directly by the designer. The detailed assembly relationships between two form features are defined as follows:

SR(FF_2, FF_1):

$$\mathbf{F}_{2i} = -\mathbf{F}_{1i} \tag{2}$$

$$\mathbf{F}_{2j} = -\mathbf{F}_{1j} \tag{3}$$

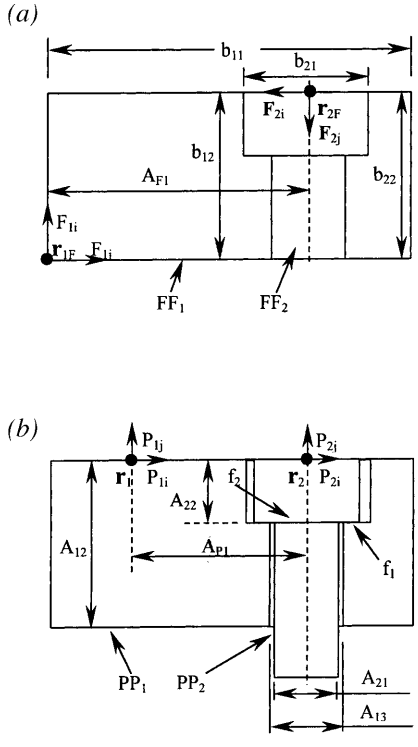


Fig. 3. Assembly relationships.

$$\mathbf{F}_{2k} = \mathbf{F}_{1k} \tag{4}$$

$$\mathbf{r}_{2F} = \mathbf{r}_{1F} + b_{22} * \mathbf{F}_{1j} + A_{F1} * \mathbf{F}_{1i} \tag{5}$$

DOF:

*Obj_has_*1*_RDOF*(FF_2, F_{2j})

The counter-bore feature can rotate about axis F_{2j}.

LTs(FF_2, FF_1):

$$A_{F1} < b_{11} - 0.5 * b_{21} \tag{6}$$

Fit (FF_2, FF_1):

$$b_{22} = b_{12} \tag{7}$$

Where

F and **r** are the orientation and position vectors of features.

$\mathbf{F}_1 = (F_{1i}, F_{1j}, F_{1k})$, $\mathbf{F}_2 = (F_{2i}, F_{2j}, F_{2k})$.

b_{ij} is the dimension of form features, Subscript i is feature number, j is dimension number.

A_{F1} is the dimension between form features.

Equations (2)–(7) present the relationships between the form feature FF_1 and FF_2. These relationships thus determine the position and orientation of a form feature in the part. Taking the part as an assembly, the form feature can be considered as "components" of the assembly.

The choice of form features is based on the shape characteristics of product-independent parts. Because the form features provided by the Unigraphics CAD/CAM system [13] can meet the shape requirements of parts for injection moulds and the spatial relationships between form features are also maintained, we choose them to build the required part models. In addition to the spatial relationships, we must record LTs, Fits relationships for form features, which are essential to check the validity of form features before updating the models in the CAD system.

3.2.2 Relationships Between Parts

In an assembly, the position and orientation of a part is usually associated with another part. Figure 3(*b*) shows a plate (PP_1) and a screw (PP_2). The relative placement of the screw is constrained by the counter-bore hole on the plate. The relationships between the screw and the plate are defined as follows:

SR(PP_2, PP_1):

$$\mathbf{P}_2 = \mathbf{M}_p \bullet \mathbf{P}_1 \tag{8}$$

$$\mathbf{r}_2 = \mathbf{M}_r \bullet \mathbf{r}_1 \tag{9}$$

SC(PP_2, PP_1):

mate(f_1,f_2) $\cap$ **axis_align**(axis_1, axis_2)

DOF:

Obj_has_1_RDOF(PP_2, P_{2j})

The screw can rotate about P_{2j} of the plate.

LTs(PP_2, PP_1):

$$A_{22} < A_{12} \tag{10}$$

Fits(PP_2, PP_1):

$$A_{13} = A_{21} + cc \tag{11}$$

Where:

$\mathbf{P}_1$ and $\mathbf{P}_2$ are the orientation vectors of the plate and the screw, and $\mathbf{P}_1 = (P_{1i}, P_{1j}, P_{1k})$, $\mathbf{P}_2 = (P_{2i}, P_{2j}, P_{2k})$.

$\mathbf{M}_p$ and $\mathbf{M}_r$ are the transformation matrix between the screw and the plate.

$\cap$ is a Boolean operator.

mate and **axis_align** are constraints (detailed discussion about them are given in the next section).

r is a position vector.

A_{ij} is dimension of parts. Subscript i is part number, and j is dimension number.

cc is the clearance between the screw and the plate.

As we can see in Eqs. (8) and (9), it is essential to calculate the matrix $\mathbf{M}_p$ and $\mathbf{M}_r$ to determine the position and orientation of the screw with reference to the plate. $\mathbf{M}_p$ and $\mathbf{M}_r$ can be derived from the spatial constraints (**SC**). This derivation requires the task of inferring the configuration of a part in an assembly, which will be discussed in the next section.

We have presented a representation of the injection mould assembly in a computer. At this stage, it might be worthwhile to summarise the benefits of this representation. Assemblies are represented as a collection of subassemblies that in turn may consist of subassemblies and/or component parts, and a component part can further be considered as the assembly of form features. Such hierarchical relationships imply an ordering on the assembly sequence and a parent–child link. The feature-based representation not only allows designers to work at a high-level of abstraction while designing individual parts, but also extends the feature paradigm to assembly modelling, because this representation allows a component to be changed parametrically with the other components consequently having their positions changed accordingly. The object-oriented representation can combine both the data structure and operation in an object. The encapsulated operational functions in an assembly object can help to automate the routine processes such as the pocketing and interference check.

4. Inferring Part Configuration in the Assembly

As we can see from Eqs (8) and (9), the positions and orientations of parts in an assembly are eventually represented by the transformation matrices. For the sake of convenience, the spatial relationships are usually specified by the high-level mating conditions such as "mate", "align" and "parallel". Thus, it is essential to derive automatically the explicit transformation matrices between parts from implicit constraint relationships. Three techniques to infer the configurations of parts in an assembly have been discussed in Section 2. Because the symbolic geometric approach can locate all solutions to constraint equations with polynomial time complexity, we use this approach to determine the positions and orientations of parts in an assembly. To implement this approach in assembly modelling software, a large amount of programming is required. Therefore, a simplified geometric approach is proposed to determine the positions and orientations of parts in an assembly.

In the symbolic geometric approach, determining positions and orientations of parts is performed symbolically by generating a sequence of actions to satisfy each constraint incrementally. The information required to satisfy each constraint incrementally is stored in a table of "plan fragments". Each plan fragment is a procedure that specifies a sequence of measurements and actions that move parts in such a way as to satisfy the corresponding constraint. The plan fragment also records the object's new degrees of freedom (DOFs) and associated geometric invariants. Conceptually, Kramer's plan fragment table is a 3D dispatch table. We use TDOF to represent translational degrees of freedom and RDOF to represent rotational degrees of freedom. Then an entry in the plan fragment table has the following form:

plan_fragment: ⟨TDOF, RDOF, constraint_type⟩
TDOF = {0, 1, 2, 3}
RDOF = {0, 1, 2, 3}
constraint_type = {coincident, in-line, in-plane, parallel_z, offset_z, offset_x and helical}

The plan fragment table is an exhaustive enumeration of all the states in the search space for the problem of moving an object to satisfy a series of constraints between markers on the object and markers fixed in the global coordinate frame. To enumerate the combination of different values of the above three parameters, 82 entries will be generated [7]. If the search space for the problem can be reduced the number of entries in a plan fragment table will decrease. To achieve this, the number of enumerate values for entry parameters must be decreased. For example, for a specified constraint type, if the enumeration values of TDOF change from {0,1,2,3} to {0,3}, then the search space is reduced.

After a careful analysis of the constraints between components of an injection mould, four basic primitive constraints are introduced: **in-line**, **parallel_z**, **parallel_z1** and **parallel_offset**. Their definitions and algebraic equations are as follows:

in-line(M_1, M_2): M_1 lies on the line through M_2 parallel to the Z-axis of M_2.

$$|[\mathbf{gmp}(M_1) - \mathbf{gmp}(M_2)]| \times \mathbf{gmz}(M_2)| = 0 \tag{12}$$

parallel_z(M_1, M_2): the Z-axes of markers M_1 and M_2 are parallel and have the same direction.

$$\mathbf{gmz}(M_1) \bullet \mathbf{gmz}(M_2) = 1 \tag{13}$$

parallel_z1(M_1, M_2): the Z-axes of markers M_1 and M_2 are parallel and have the opposite direction.

$$\mathbf{gmz}(M_1) \bullet \mathbf{gmz}(M_2) = -1 \tag{14}$$

paralle_offset(M_1, M_2, d): Applicable only in conjunction with **parallel_z** or **parallel_z1**, specifies the distance between M_1 position and M_2 position.

$$\mathbf{gmp}(M_1) - \mathbf{gmp}(M_2) = d \tag{15}$$

Where:

M_1 and M_2 are markers.

gmp(M) is the global marker position.

gmz(M) is the global marker Z-axis.

gmx(M) the global marker X-axis.

d is the distance between M_1 and M_2.

In our simplified symbolic geometric approach, the enumeration vales of constraint types are {in_line, parallel_z, parrel_z1, paralle_offset}. Compared with Kramer's symbolic geometric approach, our constraint types are reduced from seven to four. This simplification will reduce the number of entries in the plan fragment table. Based on these four primitive constraints, three high-level constraints were synthesised for the user's convenience. They are **mate** (M_1, M_2, d), **plane_align**(M_1, M_2, d), and **axis_align**(M_1, M_2). Their definitions are given as follows:

mate(M_1, M_2, d):
parallel_z1(M_1, M_2) ∩ **parallel_offset**(M_1, M_2, d)

plane_align(M_1, M_2, d):
parallel_z(M_1, M_2) ∩ **parallel_offset**(M_1, M_2, d)

axis_align(M_1, M_2):
parallel_z(M_1, M_2) ∩ **in_line**(M_1, M_2)

The assembly objects in an injection mould can have one, two or three synthesised constraints. For two and three synthesised constraints, the constraint sequence is further restricted. The sequences are as follows:

mate(M_1, M_2, d) ∩ **plane_align**(M_3, M_4, d_2)
mate(M_1, M_2, d_1) ∩ **axis_align**(M_3, M_4)
plane_align(M_1, M_2, d) ∩ **axis_align**(M_3, M_4)
mate(M_1, M_2, d_1) ∩ **axis_align**(M_3, M_4, d_2) ∩ **axis_align**(M_5, M_6).

Because of these restrictions on the constraint sequences, the number of entries in our plan fragment table is substantially reduced. To solve for one, two or three constraints allowed in our system, only nine entries are required. For interactive addition of components to the assembly, more constraint types and free sequences will increase the flexibility for users. However, in automatic assembly modelling for an injection mould, as the spatial relationships are predefined in assembly objects, some of the sequence restrictions do not matter. With the above-defined synthesised constraints, the structural relationships of a component part can be specified in the database of the components. When adding a component part to the mould assembly, the system will first decompose the synthesised constraints into primitive constraints, then generate a group of fragment plans to orient and position the component in the assembly.

5. Automated Assembly Modelling of Injection Moulds

Any assembly of injection moulds consists of product-independent parts and product-dependent parts. The design of individual product-dependent parts is based on the geometry of the plastic part [1,2]. Usually the product-dependent parts have the same orientation as that of the top-level assembly, and their positions are specified directly by the designer. As for the design of product-independent parts, conventionally, mould designers select the structures from the catalogues,

build the geometric models for selected structures of product-independent parts, and then add the product-independent parts to the assembly of the injection mould. This design process is time-consuming and error-prone. In our system, a database is built for all product-independent parts according to the assembly representation and object definition described in Section 3. This database not only contains the geometric shapes and sizes of the product-independent parts, but also includes the spatial constraints between them. Moreover, some routine functions such as interference check and pocketing are encapsulated in the database. Therefore, the mould designer must select the structure types of product-independent parts from the user interfaces, and then the software will automatically calculate the orientation and position matrices for these parts, and add them to the assembly.

5.1 Mould Base Subassembly

As can be seen from Fig. 1, the product-independent parts can be further classified as the mould base and standard parts. A mould base is the assembly of a group of plates, pins, guide bushes, etc. Besides shaping the product, a mould has to fulfil a number of functions such as clamping the mould, leading and aligning the mould halves, cooling, ejecting the product, etc. Most moulds have to incorporate the same functionality, which results in a similarity of the structural build-up. Some form of standardisation in mould construction has been adopted. A mould base is the result of this standardisation.

According to the feature-based and object-oriented assembly representation, the feature-based solid models for component parts of the mould base are first constructed; next, the assembly objects are defined by establishing relationships between components and encapsulating some functions in the component parts; then, using these assembly objects, a hierarchical subassembly object – a mould base – can be formed. This mould base object can be instantiated by a group of data from the catalogue database. Figure 4 shows the instantiation of the mould base object to generate the specified mould base. This specified mould base instance can be added automatically to the mould assembly. The structural relations between the mould base subassembly and top assembly can be expressed by Eqs. (8) and (9), where $\mathbf{M}_p$ and $\mathbf{M}_r$ are the unit matrices.

5.2 Automatic Addition of Standard Parts

A standard part is an assembly object. It can be defined according to Eq. (1) in Section 3.1. In the database, the spatial constraints are specified by *mate*, *plane_align* and *axis_align*, but unlike the mould base, the position and orientation matrices of a standard part are left unknown. During instantiation, the software then automatically infers the explicit structural relationships by using the simplified symbolic geometric approach described in Section 4.

5.3 Pocketing for Assembly Objects

One of the important issues for automatic assembly design is the automation of the pocketing process. Pocketing is an operation that makes an empty space in corresponding components to accommodate the inserted components. When an ejector is added to the assembly, an empty space is required on the EA plate to accommodate the ejector, as shown in Fig. 5.

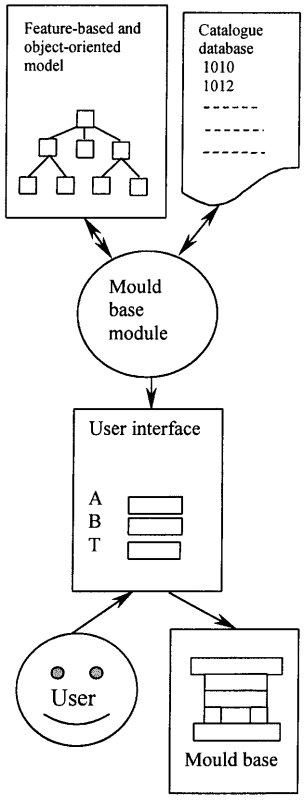


Fig. 4. Instantiation of mould base.

Since an object-oriented representation is adopted, each assembly object can be represented by two solids, the real object and the virtual object. The virtual object is modelled according to the space that a real object will occupy. Whenever an assembly object is added to an assembly, its virtual object is also added to the assembly. The operation function *pocket_plate()* in **M** of **O** will subtract the virtual object from the corresponding components (see Eq. (1) and Table 1). Moreover, because there are associativities between the virtual object and real object, the pockets on the corresponding components will change with the modification of the real object.

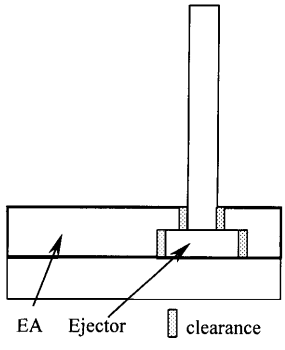


Fig. 5. Pocketing of assembled objects.

This automatic pocketing function further demonstrates the advantage of an object-oriented representation.

6. System Implementation

Based on the Unigraphics system [13], the proposed feature-based and object-oriented assembly scheme and automation of assembly modelling have been implemented in the IMOLD system [14] developed at the National University of Singapore. The Unigraphics system provides a user-friendly application programming interface (API). Through this interface, the users can call Unigraphics internal functions such as adding parts to an assembly, modifying parameters, etc. Although Unigraphics provides functions for mating conditions, the proposed approach is still needed to infer the component configuration, because it is necessary to calculate the degrees of freedom, and check the validity of mating conditions before the component can be added to the assembly. The proposed synthesised constraints are compatible with Unigraphics constraints.

Figure 6 shows an injection moulded product, and the designed injection mould assembly for this product is shown in Fig. 7(*a*). The corresponding parent–child relationships for fix-half subassembly are shown in Fig. 7(*b*). This assembly is designed by the IMOLD system. Each plate of the mould base is automatically positioned in the assembly. The standard parts such as the locating ring and ejector are added to the assembly automatically, and the pockets for these standard parts are also created automatically.

7. Conclusion

The proposed feature-based and object-oriented hierarchical representation for injection mould assembly not only extends the feature paradigm to the assembly design, but also encapsulates operational functions and geometric constraints, such as the degree of freedom, mating conditions, insertion and orientation limits, etc. Because of the extension of the feature paradigm to assembly design, the modifications, such as dimensional changes of the assembled component can be made even after the completion of the assembly process. The encapsulation of assembly objects has the following two advantages: first, because the mating conditions are encapsulated in the assembly objects, automatic assembly design is easy to implement; secondly, the encapsulated operational functions in the object assembly automate the routine processes of assembly design, such as pocketing and interference check. The proposed simplified action analysis can substantially reduce the programming effort needed for automatic checking for component interference within a mould assembly.

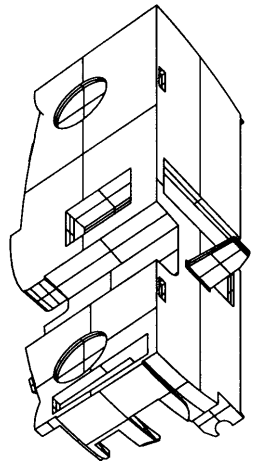


Fig. 6. An injection moulded product.

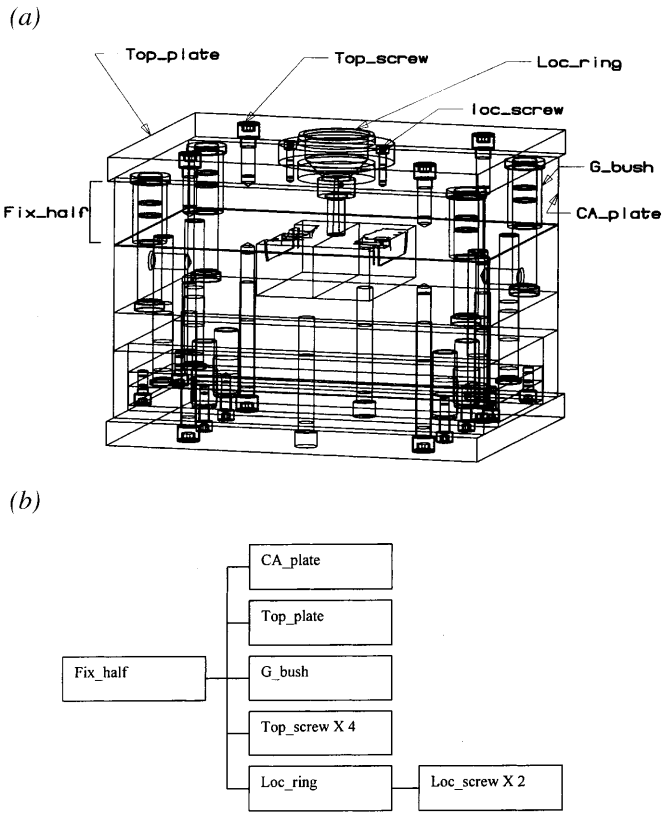


Fig. 7. The mould assembly for the product in Fig. 6.

Acknowledgement

The authors would like to thank the National University of Singapore and National Science and Technology Board of Singapore for supporting the IMOLD® [14] research project.

References

1. K. H. Shin and K. Lee, “Design of side cores of injection moulds from automatic detection of interference faces”, Journal of Design and Manufacturing, 3(3), pp. 225–236, December 1993.
2. Y. F. Zhang, K. S. Lee, Y. Wang, J. Y. H. Fuh and A. Y. C. Nee, “Automatic slider core creation for designing slider/lifter of injection moulds”, CIRP International Conference and Exhibition on Design and Production of Dies and Moulds, pp. 33–38, Turkey, 19–21 June 1997.

3. E. C. Libardi, J. R. Dixon and M. K. Simmon, "Computer environments for design of mechanical assemblies: A research review", Engineering with Computers, 3(3), pp. 121–136, 1988.
4. K. Lee and D. C. Gossard, "A hierarchical data structure for representing assemblies", Computer-Aided Design, 17(1), pp. 15–19, January 1985.
5. K. Lee and D. Gossard, "Inference of position of components in an assembly", Computer-Aided Design, 17(1), pp. 20–24, January 1985.
6. A. P. Ambler and R. J. Popplestone, "Inferring the positions of bodies from specified spatial relationships", Artificial Intelligence, 6, pp. 157–174, 1975.
7. G. Kramer, Solving Geometric Constraint Systems: A Case Study in Kinematics, MIT Press, Cambridge, MA, 1992.
8. J. J. Shah and M. T. Rogers, "Assembly modelling as an extension of feature-based design", Research in Engineering Design, 5(3&4), pp. 218–237, 1993.
9. J. P. Kruth, R. Willems and D. Lecluse, "A design support system using high level mould objects", CIRP International Conference and Exhibition on Design and Production of Dies and Moulds, pp. 39–44, Turkey, 19–21 June, 1997.
10. J. J. Shah, "Assessment of feature technology", Computer-Aided Design, 23(5), pp. 331–343, June 1991.
11. S. R. Gorti, A. Gupta, G. J. Kim, R. D. Sriram and A. Wong, "An objection-oriented representation for product and design process", Computer-Aided Design, 30(7), pp. 489–501, June 1998.
12. J. Rumbaugh, M. Blaha, W. Premerlani, et al. Object-Oriented Modeling and Design, Prentice Hall, Englewood Cliffs, NJ, 1991.
13. Unigraphics Essentials User Manual, Unigraphics Solution Co., Maryland Heights, MO, 1997.
14. IMOLD homepage http:://www.eng.nus.edu.sg/imold, Manusoft Plastic Pte Ltd. Singapore.