Int J Adv Manuf Technol (2007) 32: 434–444
DOI 10.1007/s00170-005-0371-8

ORIGINAL ARTICLE

Y.-S. Ma · G. A. Britton · S. B. Tor · L. Y. Jin

Associative assembly design features: concept, implementation and application

Received: 12 January 2005 / Accepted: 10 October 2005 / Published online: 12 May 2006

Abstract This paper presents a new type of feature, associative assembly design feature. Its concept, implementation, and application are introduced. This new feature allows the following associations: (1) between parts that have not been defined geometrically; (2) between geometric entities defining interfaces between parts; and (3) between part geometry and intermediate geometry used to define a part. The associations can be geometric or non-geometric. Our extension to traditional assembly feature properties allows product architectures to be defined using features. These architectures, in turn, can be used to constrain the modular design of assembly geometry. The application is a mould base library for injection mould design.

Keywords Assembly feature · Associative feature · Feature-based design · Design library

1 Introduction

CAD systems have become smarter through embedding expert knowledge and linking with specialist engineering solutions. This trend will continue as more powerful expert systems are developed for engineering design and verification. The major challenge is to develop a 'design-centric' expert system that incorporates knowledge from all phases of the product life cycle and supports global, concurrent design and engineering. The transference of information from CAD to knowledge-based engineering (KBE) systems is very difficult because KBE systems rely heavily on design intent to perform activities such as DFX analyses and cost estimation [4]. The intelligence added to CAD geometry is either stripped off by the translation software or not recognized by the KBE system. In addition, many current CAD systems cannot capture design intent completely and unambiguously. On the other hand, transferring KBE intelligence to CAD systems is equally challenging because there is no mechanism to enable such information flow. The gap between CAD and KBE systems can be bridged using feature-based engineering [12, 14].

In this paper, the authors propose a new type of feature, associative assembly design feature, and describe its concept, implementation and an application with a mould base assembly library. Section 2 reviews research in the field of feature modelling. Section 3 introduces the associative feature concept and its characteristics. Section 4 discusses the concept and implementation of associative assembly design features. To show the effective application of this new feature type, Section 5 presents an assembly library framework designed by using assembly features. Section 6 discusses the detailed library implementation and demonstrates the feature editing functions. Finally, Sect. 7 contains the summary and conclusions.

2 Literature review on assembly features

In most CAD/CAM tools, machining form features are emphasized so that part modelling, process planning, and CAM tool path generation for machined parts can be interfaced using common geometric information. Theoretically, feature-based technology should offer consistent data sharing among other application domains such as conceptual design, assembly planning, and quality inspection, but implementation is very difficult due to the differences in feature definitions. Current practice is limited to extracting or copying CAD geometry entities from one to another application, instead of dynamically linking them. Limited integration has been achieved by mapping between design and other engineering features; for example, the CAD and CAE feature integration carried out by Deng et al. [5].

Y.-S. Ma (✉) · G. A. Britton · S. B. Tor · L. Y. Jin
School of Mechanical and Aerospace Engineering,
Nanyang Technological University,
50 Nanyang Avenue,
639798 Singapore, Singapore
e-mail: mysma@ntu.edu.sg
Tel.: +65-67905913
Fax: +65-67911859

Shah and Rogers [15] developed feature-based design for assemblies. They provided a uniform set of connection relations for modelling all levels of the product: part-of, structuring relations, degrees of freedom, motion limits, and fit. Such features can be used for kinematics analysis, tolerance analysis, and assembly analysis and planning. One very interesting point is that they distinguish between the representation of the assembly and that of assembly relationships. In our view this de-coupling is essential for assembly features. Unfortunately their declarative approach has limited capability for associating geometric properties. Their form features are volumetric, requiring pre-definition of part geometry. Such feature-in-part constraints limit the assembly features that can be defined.

More recently, connection features between pre-defined geometric entities have been used to define the geometric positions, orientations, mating conditions, and parent-child relations [3, 16, 22]. Connection features are essential for assemblies but they are very specialised. Henson et al. [7] argue that a product model should include rich information such as component geometry, assembly sequences, mating relations, representation of functions, cost and type of assembly process, costs of component manufacture, and component differentiation, integration and design rules, etc. This view is echoed by Holland and Bronsvoort [8], who introduced the concept of manufacturing set-up handling features in addition to connection features in assemblies. However, their mixed part-assembly approach is not compatible with the basic principles of embodiment design in mechanical engineering, although it does facilitate information sharing within a generic analysis model.

Traditionally a product model consists of a hierarchical framework-product, assembly, components, and features, each of which has a set of entity attributes [7, 21]. The assembly feature semantics defined under such a framework are useful for many bill-of -material-based applications, such as drawing creation, manufacturing, assembly, scheduling, inventory control, and maintenance services. It is not suitable for those applications that require semantic information about the relationships between entities across components or assemblies, such as DFX engineering analysis and optimisation [7, 15]. For example, to reduce assembly cost, a design model may be optimized to reduce the number of parts [7]; therefore, assembly design features can not be limited by the boundaries of part solids.

More importantly, the previous assembly feature definitions require detailed form features defined in part models. They support bottom-up development, but offer no solution for top-down development; so they are totally inadequate for the early stages of design when key design decisions are being made about the architecture. Architectural design requires design constraints to be specified between *named* parts or components, for which the part (volumetric) geometry does not exist (because they have not been designed yet). In this situation the traditional connection features cannot be used. A partial solution to this problem was developed by Myung and Han [11], who introduced a "design unit" concept, which is a sub-assembly model that attaches functional design features with assembly-part groups, but they did not provide the means to edit and manage assembly features. Another solution is to use a functional design approach. One popular functional design method is the function-behaviour-structure approach [2, 6, 13, 17, 23]. Functional design has an important contribution because it provides a clear link between design intention and physical structure. Thus far, work in this field has focussed on conceptual design and the integration with detail design is immature.

Recently the feature concept has been extended to include any meaningful grouping schema related to geometrical entities [12]. Betting and Shah [1] suggest a cross-application feature based architecture. Brunetti and Golob [3] state that a major prerequisite for a schema is a suitable representation scheme to store, manage, and retrieve product semantics including conceptual data. Ma and Tong [9] argue that features have to be flexible, self-contained, and consistent to integrate different applications for concurrent engineering.

From the brief discussion it can be seen that there are still difficulties in adequately defining all the characteristics required of features for detail design. The authors argue that the semantics of an assembly feature should include the following:

- Independent representation of the part-assembly relations and the design pattern relations among parts in the assembly
- Representation of relationships between named features of named parts, even before their geometry has been defined. This is especially important during the early design stages
- Representation of connections between defined geometric entities
- Representation of both geometric and non-geometric relations in an assembly
- Provision of means to interface with different engineering applications

The authors conclude that these conditions have not yet been fulfilled. This paper presents our scheme to satisfy these conditions. The key concept in the scheme is *associative feature*.

3 Associative feature

Ma and Tong [9] introduced the concept of associative features (AFs) to support the integration of concurrent engineering. An associated feature is defined as a set of semantic relationships among product geometric entities, which can be defined as a single object entity in an engineering application. The product geometric entities may include assemblies, components, solids, faces, edges, vertices, surfaces, curves, points, vectors, datum references, etc. The relations can be geometric or non-geometric, including constraints, dependencies, equations, memberships, part-whole relations, coupling, patterns, etc.

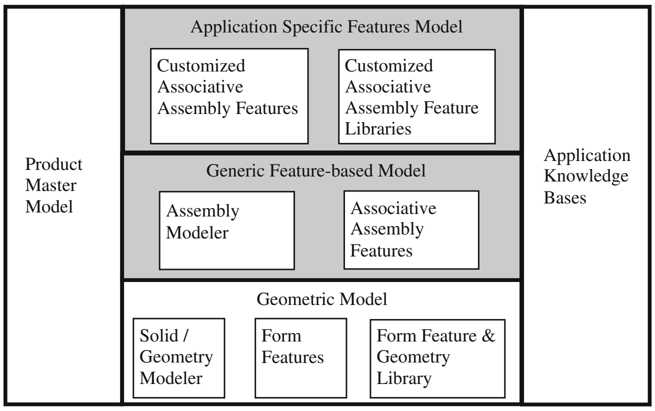

Application Specific Features Model
Customized Associative Assembly Features
Customized Associative Assembly Feature Libraries
Generic Feature-based Model
Assembly Modeler
Associative Assembly Features
Geometric Model
Solid / Geometry Modeler
Form Features
Form Feature & Geometry Library
Product Master Model
Application Knowledge Bases

Fig. 1 Layer structure of associative assembly features

An associative feature has the following key characteristics:

– Built-in associative links to the related geometric entities
– Self-validation to check for the consistency of its entities, attributes, constraints, etc.
– Methods for constructing, storing, indexing, editing and destroying its instances
– Methods that can be expanded to interface with query and execution mechanisms for high level knowledge processes
– Methods to interface with other engineering application tools

In addition to the characteristics listed above, AFs can be distinguished from traditional geometrical form features in the following aspects:

– AFs are objects defined specifically for an application purpose.
– AFs can be defined independently of geometric model entities.
– AFs refer to product final geometric model entities, but they are not derived from them. For example, an associative feature may contain intermediate geometry needed to construct another geometric entity, to which it is referenced.
– An AF does not require a volume of material like a form feature; therefore, the feature elements can refer to any geometric entity, including datum entities. This is the key feature of the AF definition. It allows geometric and other design patterns to be specified prior to the detailed design and geometric definition; for example, design patterns for a plastic injection mould might include cooling circuit layouts, core/cavity insert layouts, sub-insert layouts, hole patterns for mould plates, and gate and runner circuit layouts [9].
– Volumetric entities such as form features, assemblies and components are special types of AFs. Note that traditional machining features are included in our definition.
– AFs can refer to derived entities defined with respect to the final product geometry model. Geometric solid representation and associated relevant rules are optional; they are only applied when necessary.

The architecture for our proposed associative assembly design feature system is shown in Fig. 1. The bottom layer contains the solid geometry modeller, form features, and form feature and geometry libraries. These functions are widely available today. Our contribution lies in the two layers above, shown shaded in Fig. 1. The layer above the geometric modeller consists of generic associative assembly features and an assembly modeller. The associative assembly features integrate with and extend the functionality of conventional assembly modellers, which use connection features. The top layer is the engineering application layer. At this level, generic features can be customised to create domain specific associative assembly features and feature libraries. Along the sides of these three layers are the product master model and application knowledge bases. These interact with all three layers. Our associative feature semantics facilitates interactions with the product master model and engineering knowledge bases.

4 Assembly design feature

The main purpose of this paper is to introduce a new category of associative features, assembly design features. In our proposed associative feature model, associative assembly relations are classified according to purpose, for example, assembly (functional) design, engineering analysis, assembly planning (sequences), manufacturing pro-

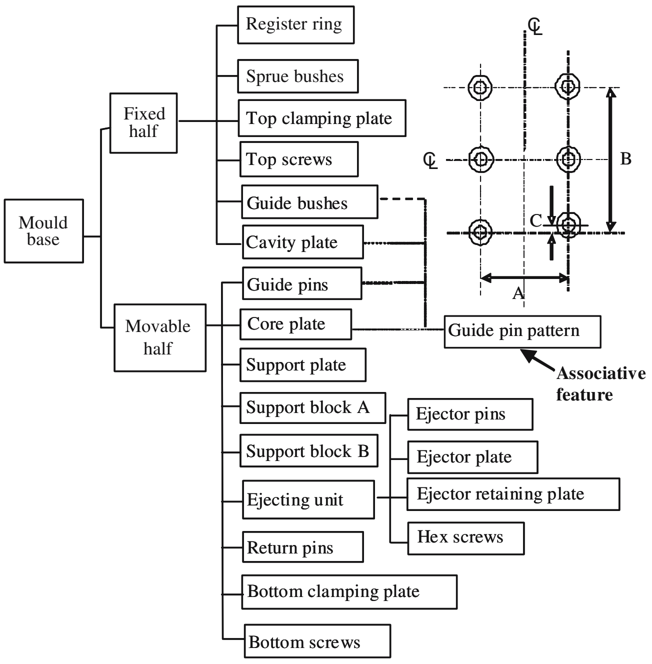


Fig. 2 An assembly design feature related to a mould base assembly structure

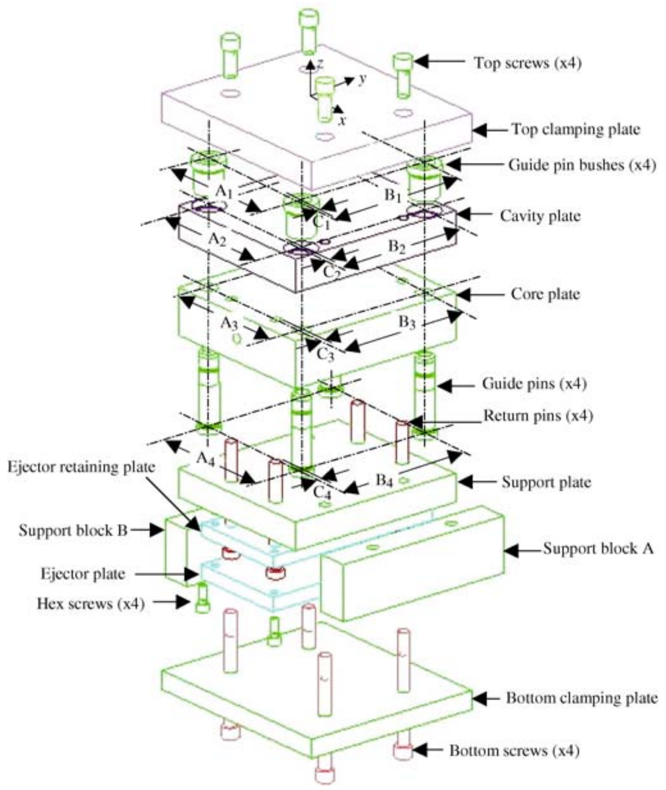
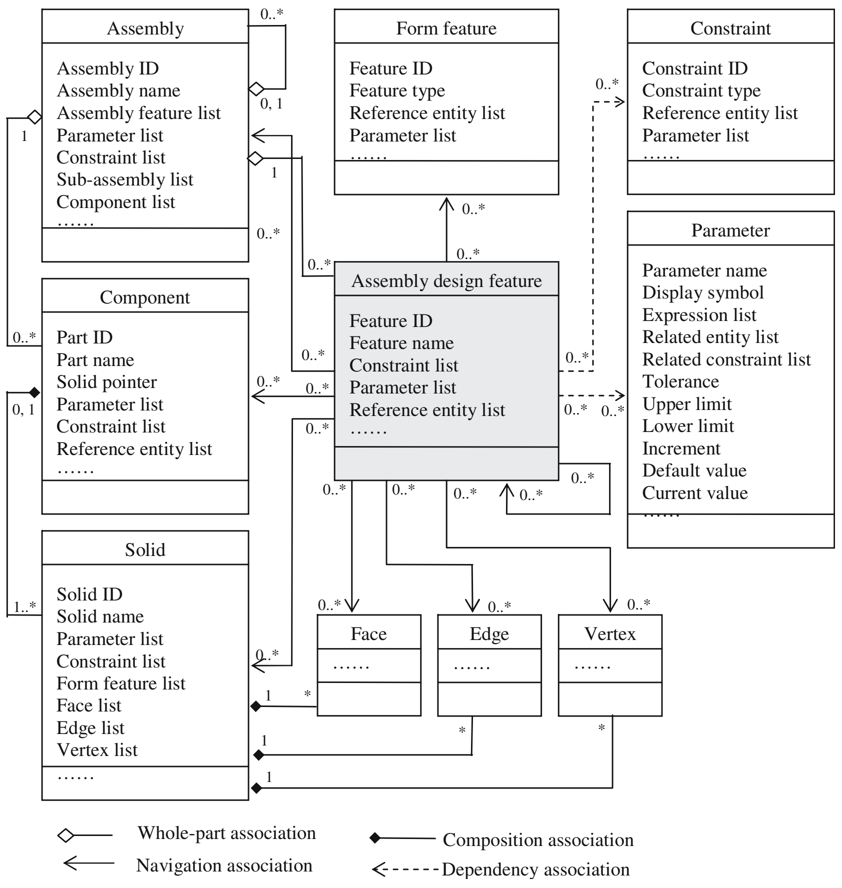


Fig. 3 Illustration of an assembly design feature in a geometric assembly model

cess planning, etc. This paper addresses assembly design features for modular systems. A typical example is illustrated in Figs. 2 and 3 for a plastic injection mould base assembly. In this case, the assembly design feature is a pattern defining the number and positions of the guide pins. For convenience the parameters controlling the pattern are labelled *A*, *B*, and *C*.

Figure 2 shows a typical assembly tree for a mould base. The mould base assembly is divided into three subassemblies, namely, fixed subassembly, movable subassembly, and ejection subassembly. Each subassembly is made up of components. The relationships shown in the assembly tree are based on part-whole relations. In the geometric model, it is possible to include mating relationships between the components and the sub-assemblies. This capability is already widely available in commercial CAD software. However, in addition to these types of relationships there are other dimensional relationships that apply to a number of components in the assembly. The guide pin pattern is one example. This kind of pattern applies to several different components in different sub-assemblies, some of which are not in direct contact with each other: guide bushes, cavity plate, guide pins and core plate. In addition, it is desirable to store this pattern independently of the related components so it can be used during conceptual design. This is achieved through our associative feature semantics. There are three points to note. First, it is normal practice to off-set one pin to ensure that the mould can be

Fig. 4 Partial relations defined in the assembly design feature class



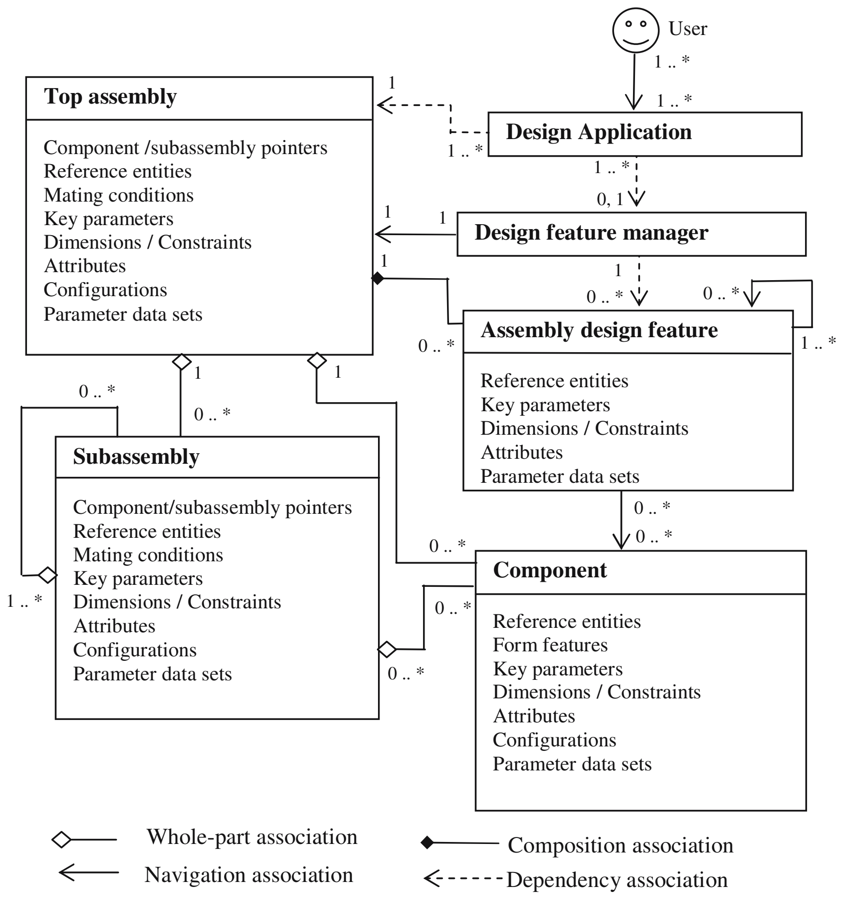


Fig. 5 The application relations created by assembly design features with CAD assemblies and components

assembled in only one way. Second, the pattern in Fig. 2 shows six guide pins, but the mould in Fig. 3 only has four pins. The number of pins depends on the mould size. The associate feature defining this pattern contains rules to determine the number of pins required for a specific mould size.

The third point is very important. The pattern is defined parametrically. In this case, the dimensions are indicated by the parameters A, B, and C. These parameters are linked to the relevant parameters for the components controlled by the pattern. The manner in which this is achieved is illustrated in Fig. 3, which shows an exploded assembly model of the mould assembly.

The components inherit parameters from the guide pin pattern at the mould assembly level. The guide bush holes in the cavity plate and those in the core plate have the same pattern as the predefined assembly design feature. In the cavity plate part, A_2, B_2 and C_2 are used to define the local pattern of the guide pin bush holes, which are, in turn, parametric and primitive form features. Similarly, A_3, B_3 and C_3 are used in the core plate part.

The guide bushes and guide pins are directly related to the cavity and core plates through mating conditions. The mating conditions, in turn, are defined with respect to their accommodating holes in the core and cavity plates. That is:

Position of bush (or pin) = position of hole in cavity (or core) plate + offset from hole where, the offset is determined by the mating condition.

Hence, the positions of the bushes and pins, (A_1, B_1 and C_1) and (A_4, B_4 and C_4), are associated with and controlled by the guide pin pattern.

It is important to note that the naming of parameters for the components can be, and in general will be, different from the naming used in the design feature pattern. The assembly design feature object provides the semantic linking between the component and pattern parameters.

Figure 4 shows a partial class relation diagram in UML format for an *assembly design feature*. The properties include reference entity list, constraint list, parameter list, etc. Related geometric entities can be assemblies (including its parent assembly), components, solids, form features, faces, edges, vertices, etc. They can also be derivatives of these geometric entities or construction datum references. Assembly design features can also refer to other assembly design features.

The application of assembly design feature can be achieved as shown in Fig. 5. The assembly design feature is implemented as an object class in the design application environment. During the design session, the class is loaded and ready for instantiation. A feature manager is needed to register, track and check consistency of all the design features created in the session. The properties of each object are associated with the top assembly, sub-assemblies or components. The associations can be achieved in different ways. The method proposed in this work will be introduced in the next section.

The novelty of the approach lies in the concept and implementation of the assembly design feature object and its associated feature manager. In general, the associations are implemented using CAD API's, which are embedded in the methods of the assembly design feature object.

The user can use the assembly design feature in either a top-down or bottom-up approach. In the top-down approach, the designer can develop the conceptual design based on the results of a function mapping process. The conceptual design can be further refined as design patterns in either geometric or non-geometric representation. The design patterns can then be converted to assembly design features to define the product architecture. Alternatively, predefined patterns can be selected from an assembly features library. In the bottom-up approach components and form features that have already been defined can be associated using the assembly design feature.

It can be appreciated that in the context of product modelling with assemblies, many other different associative features can be defined and applied for different purposes such as assembly planning, manufacturability analysis, cost evaluation, process planning, quality checking, etc. It is possible to generalise associative features to cover all kinds of assembly purposes. We refer to these as associative assembly features (AAFs). An assembly design feature is one type of AAF.

4.1 Implementation of assembly design features

Assembly design features have been implemented in QuickMould using Unigraphics CAD software (Fig. 6). The left side of the figure shows that assembly design features contain the following properties: assembly parameters, assembly parametric constraints, position and orientation constraints, assembly configuration, and predefined sizes.

On the right side of Fig. 6, there are three layers. The middle layer consists of all the entities contained in the CAD assembly model: a top assembly part, a few subassembly parts and component parts. The parts contain expressions, which are automatically generated and used for parametric modelling. There are two types of expressions: inter- and inner-part. Inner-part expressions are associated with the modelling entities contained in the same part, while inter-part expressions are associated with the expressions of other parts. The assembly tree is defined in an assembly file.

Assembly design feature properties are associated with the CAD entities as shown in Fig. 6. For example, assembly parameters are associated with top assembly expressions, which are contained in the top assembly part. Feature position and orientation constraints are associated with the assembly mating conditions defined at a different node of the assembly tree.

The bottom layer on the right contains the parameter data sets for different product configurations and sizes. The predefined sizes for a parametric family are defined by sets of predefined parameter values. Thus, modular design configurations for product families can be standardized and "frozen" for future reuse. Note that this approach covers both scale-based and modular-based product families.

The top layer on the right of Fig. 6 contains the configuration files and bitmap files for the feature manipulation function UI's. The bitmaps provide a picture of the feature in the UI for ease of use. This layer is needed

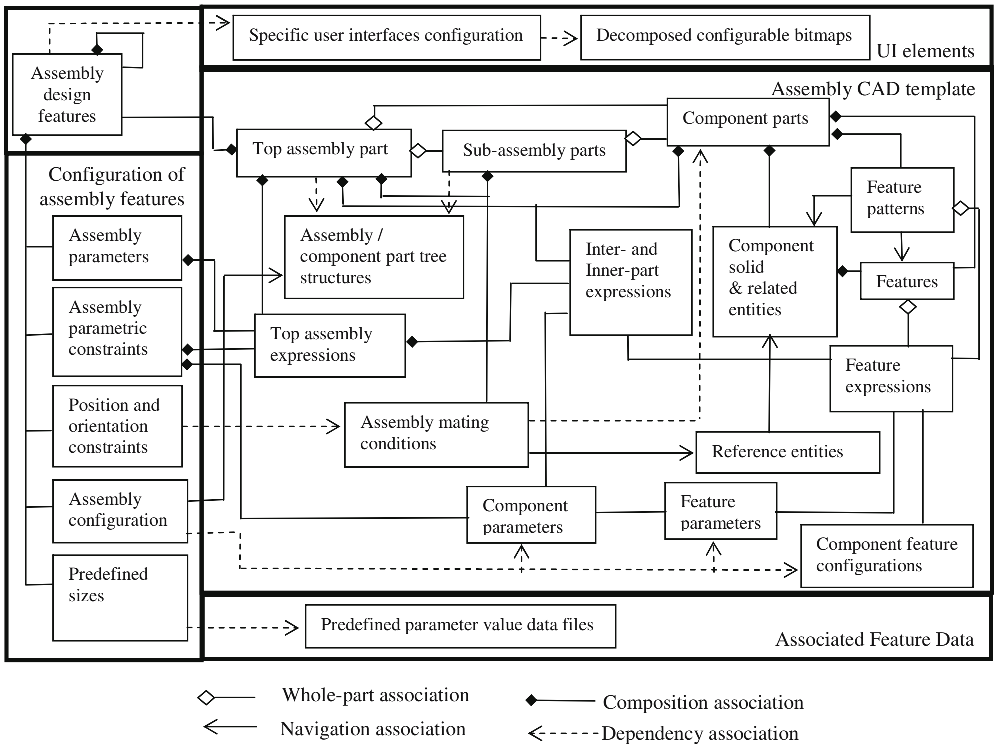


Fig. 6 Implementation of the assembly design features

because the generic assembly features can be customised for different applications. For each application customised assembly design features can be defined. Pictures of these features can be defined using decomposed configurable bitmaps and embedded in specific user interfaces.

The parametric association between component part feature parameters and assembly design feature parameters have been implemented using CAD software API's that define association rules in terms of expression functions. An assembly design feature (left-top corner) contains parameters (Fig. 6). These design feature parameters are expressed in the assembly expressions at different levels. In Fig. 2, the "guide pin pattern" feature parameters are expressed at the top-assembly part level. Features and feature pattern expressions in components or child-assembly parts can be associated with the parent assembly expressions via inter-part expressions (associations). Note, however, that this method of implementation is for convenience; our feature concept is not limited solely to expression functions.

The procedure is illustrated with the assembly design feature shown in Fig. 2, where the pattern variations were defined by different values of *A*, *B* and *C*. For example, the number of instances (number of guide pins) along the *y* direction increases when the value of B becomes large. In the implementation, the actual names of *A*, *B* and *C* are "gp_dis_x", "gp_dis_y", and "gp_h_offset_y", respectively. To enable parametric association with the design pattern, the components must be defined using parametric features. Figure 7 shows some of the parameters defined for the "cavity plate" of Fig. 3.

The key parameters "gp_dis_x", "gp_dis_y", and "gp_h_offset_y" are mapped as expressions "mb2_a_s_gp_dis_x", "mb2_a_s_gp_dis_y" and "mb2_a_s_guid_pin_offset_y" in the top assembly part "mb2_a_s" (Fig. 8a). Next, at the "cavity plate" component level the generic parameters "gp_dis_x", "gp_dis_y", and "guide_pin_offset_y" are linked to the cavity plate parameters "pa_gp_h_dis_x", "pa_gp_h_dis_y" and "pa_gp_h_offset_y" using inter-part expression references, which follow the convention: **"local parameter = reference part :: reference parameter"**, e.g., "pa_gp_h_dis_x = mb2_as :: mb2_a_s_gp_dis_x". Figure 8b shows a portion of the corresponding expressions defined

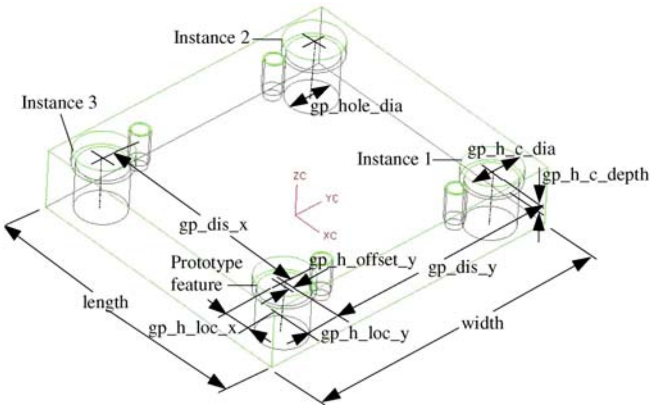


Fig. 7 A component model with parameters shown (cavity plate)

for the component part. Other features are defined in a similar manner.

An interesting variation on the parameter definition is the situation where the number of instances is made to vary depending on the overall size of the assembly. Consider the case where the number of guide pins and bushes increases with mould size. If, in the *y* direction, there are "gp_n_y" instances, and this expression has been defined in the top assembly, then the pattern related component expressions in "pa" ("plate A" part, which refers to the cavity plate) would be:

"pa_gp_n_h_y"=mb2_as::mb2_a_s_gp_n_y"
"pa_gp_h_dis_y=mb2_as::mb2_a_s_gp_dis_y/ (pa_gp_n_h_y-1)"

As the number of instances is increased, the number of guide pins and bushes, and their accommodating holes are created parametrically.

Other methods have been implemented for assembly design features to provide functions over the complete design life cycle. These include initiation, enrichment, modification, information queries, deletion, etc.

To configure the design environment, the types of assembly design features have to be registered first. When creating a feature object, relevant pointers can be obtained by interactive entity selection or construction in the design environment. Once the object is fully established, by embedding the relevant properties into the design model entities, it can be persistently stored in the product model. The feature object can be edited via a unified user interface coupled with common call-back functions. Then the new object contents can be saved again and again. At the beginning of a design session it is necessary to cluster all the assembly features that exist in the working assembly. This is achieved by cycling for the earmarked attributes until the matching entities are identified. Then the feature objects are established by retrieving the embedded attributes.

5 Design of a mould base assembly library

The mould base design approach suggested here is in line with the assembly design theory presented by Whitney et al. [18–20]. In their papers, it was shown that assemblies are structures of mutual constraints instantiated by assembly features. These structures build chains of parts that can be represented by a directed acyclic graph called a datum flow chain.

In this work, a comprehensive mould base library has been implemented with assembly design features. Standard mould base catalogues are classified according to different suppliers, even though they have different feature definitions and dimensional conventions. Each catalogue specifies sets of standard mould base types and components. For each standard configuration, an assembly CAD template is built together with a configuration file and a data file as shown in Fig. 9. The configuration file specifies the assembly features within the template and their corresponding parameters, dimensions, and other attri-

Fig. 8 Inter- and inner-expressions of component and feature parameters

```
//Assembly level expressions for guide pin pattern
mb2_a_s_gp_dis_x=142        // Guide pin pattern parameter A
mb2_a_s_gp_dis_y=154        // Guide pin pattern parameter B
mb2_a_s_guid_pin_offset_y=2 // Guide pin pattern parameter C
... ...
```

a

```
// cavity Plate component level expressions
//connected to the corresponding parent expressions
pa_gp_h_dis_x=mb2_as::mb2_a_s_gp_dis_x                   // parameter A
pa_gp_h_dis_y=mb2_as::mb2_a_s_gp_dis_y                   //Parameter B
pa_gp_h_offset_y=mb2_as::mb2_a_s_guid_pin_offset_y       // parameter C
... ...
```

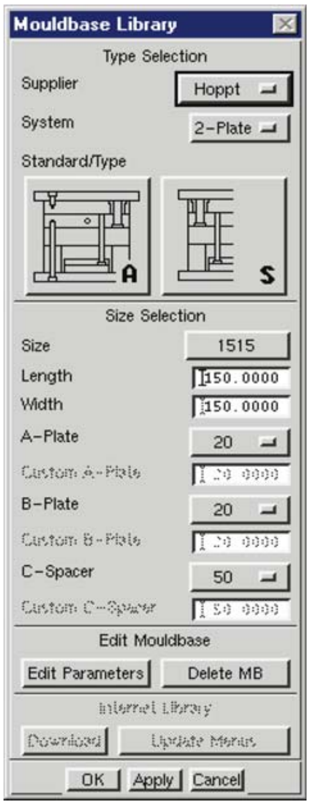
b

butes. They can be flexibly configured to reflect different suppliers' definition conventions. Once the user confirms a selected standard, an instance of the assembly is inserted into the design model and its corresponding assembly features are registered by a feature manager in the design environment (see Fig. 5), which in turn associates the assembly instance with the corresponding library source via a library manager. This feature manager is also responsible for maintaining the validity of assembly features through a set of general methods, such as "*select_sizes()*", "*update_parameters()*" and "*evaluate_-constraints()*", etc.

6 Library implementation

The system was developed based on and seamlessly integrated with, UG via UG/OPEN API using C++ language. When the library is invoked in the QuickMould environment, the program automatically initialises the session and checks for available catalogues, standards, and types, as well as the corresponding configurations, data and template files. Then the library resource tree is established which in turn supports the mould base library main UI as shown in Fig. 10. The user can select different suppliers from a top pull-down menu, e.g. DME, FUTABA, and HOPPT, and the required configuration out of the available options, such as 2- or 3-plate mould bases. In each category, mould sub-types can be selected as shown in Fig. 11 for DME 2-plate mould bases.

When the "Apply" or "OK" button is activated at the bottom of the UI the selected mould base assembly is loaded and inserted into the design assembly, as shown in Fig. 12. In this case, the default size is "1515" which means 150 mm in length and 150 mm in width. When the user clicks the "size" pull-down menu all the available default sizes for the specific configuration is listed in another sub-menu. When the user selects a new size, it does not load



Supplier	Hoppt
System	2-Plate
Size	1515
Length	150.0000
Width	150.0000
A-Plate	20
Custom A-Plate	20.0000
B-Plate	20
Custom B-Plate	20.0000
C-Spacer	50
Custom C-Spacer	50.0000

Fig. 10 The main UI of the mould base library in QuickMould

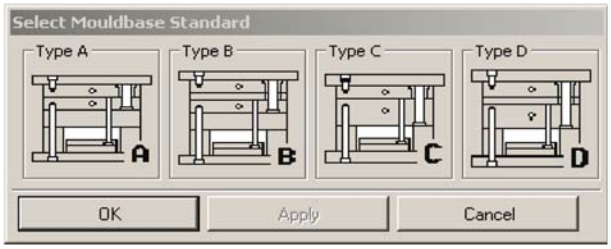
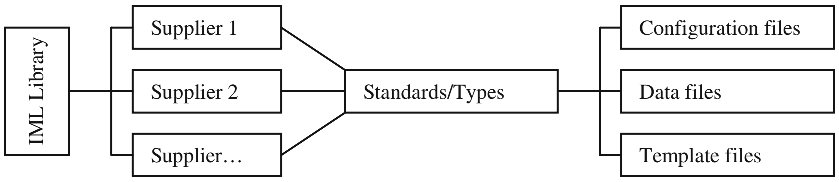
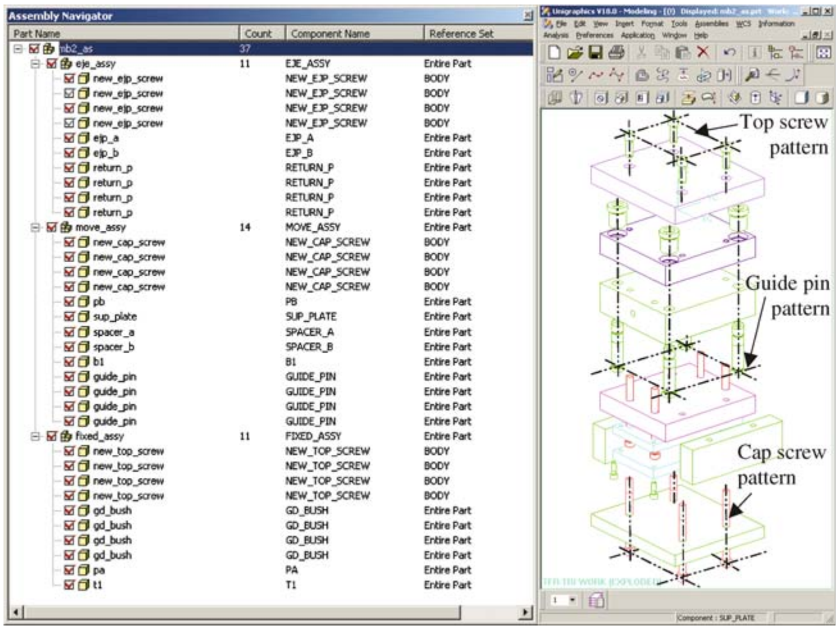


Fig. 11 The selection UI of 2-plate DME mould base standards

Fig. 9 IML library data structure





Assembly Navigator

Part Name	Count	Component Name	Reference Set
mb2_as	37		
eje_assy	11	EJE_ASSY	Entire Part
new_ejp_screw		NEW_EJP_SCREW	BODY
new_ejp_screw		NEW_EJP_SCREW	BODY
new_ejp_screw		NEW_EJP_SCREW	BODY
new_ejp_screw		NEW_EJP_SCREW	BODY
ejp_a		EJP_A	Entire Part
ejp_b		EJP_B	Entire Part
return_p		RETURN_P	Entire Part
return_p		RETURN_P	Entire Part
return_p		RETURN_P	Entire Part
return_p		RETURN_P	Entire Part
move_assy	14	MOVE_ASSY	Entire Part
new_cap_screw		NEW_CAP_SCREW	BODY
new_cap_screw		NEW_CAP_SCREW	BODY
new_cap_screw		NEW_CAP_SCREW	BODY
new_cap_screw		NEW_CAP_SCREW	BODY
pb		PB	Entire Part
sup_plate		SUP_PLATE	Entire Part
spacer_a		SPACER_A	Entire Part
spacer_b		SPACER_B	Entire Part
b1		B1	Entire Part
guide_pin		GUIDE_PIN	Entire Part
guide_pin		GUIDE_PIN	Entire Part
guide_pin		GUIDE_PIN	Entire Part
guide_pin		GUIDE_PIN	Entire Part
fixed_assy	11	FIXED_ASSY	Entire Part
new_top_screw		NEW_TOP_SCREW	BODY
new_top_screw		NEW_TOP_SCREW	BODY
new_top_screw		NEW_TOP_SCREW	BODY
new_top_screw		NEW_TOP_SCREW	BODY
gd_bush		GD_BUSH	Entire Part
gd_bush		GD_BUSH	Entire Part
gd_bush		GD_BUSH	Entire Part
gd_bush		GD_BUSH	Entire Part
pa		PA	Entire Part
t1		T1	Entire Part

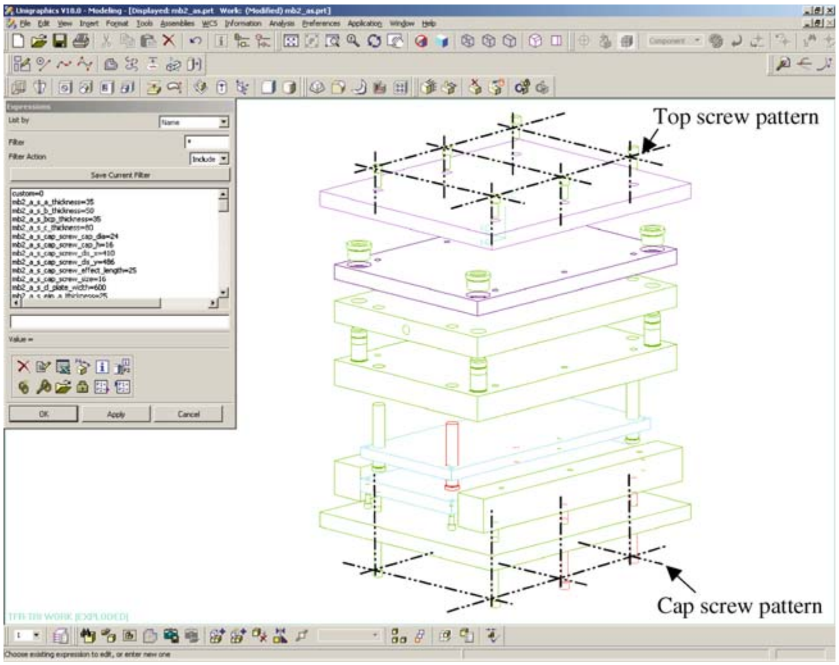
Fig. 12 The CAD model of 2-plate A standard mould base from DME

another set of assembly parts because there are many different sizes in every configuration. If each size is implemented in a "hard-coding" manner, the mould base library CAD files would be enormous. Then such a library would be too large to manage. Our design assembly feature provides a generic method to parametrically update and regenerate the existing mould base assembly with the corresponding data set, which specifies the effective parameter values predefined for the selected size by the supplier. Figure 13 shows the updated mould base assembly in a new size. Comparison between Figs. 12 and 13 shows that the updated mould contains more screws. This clearly demonstrates that our method can handle subtraction and addition of features and the related components, an essential feature for CAD modelling of modular product families.

The novelty here is that, in this system, assembly design features such as the thickness values of plates, the cap screw pattern, and the guide pin pattern are modelled, identified and managed as objects.

Within a library assembly template, associative parametric links are automatically created via relations among expressions across the assembly model. Such links are retrieved, managed and saved via a set of feature object modification methods.

A user can customize a mould base assembly easily by editing the assembly design features. For example, if the designer decides that a "2×2" guide pin pattern is not



Unigraphics V18.0 - Modeling - [Displayed: mb2_as.prt Work: (Modified) mb2_as.prt]

Expressions

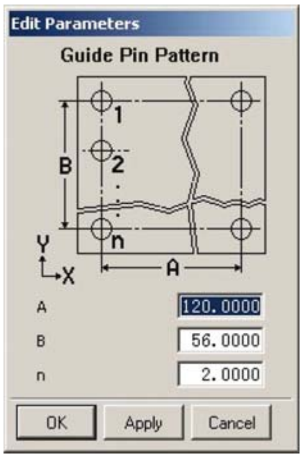
List by: Name
Filter: *
Filter Action: Include

```
custom=0
mb2_a_s_a_thickness=35
mb2_a_s_b_thickness=50
mb2_a_s_bcp_thickness=35
mb2_a_s_c_thickness=80
mb2_a_s_cap_screw_cap_dia=24
mb2_a_s_cap_screw_cap_h=16
mb2_a_s_cap_screw_ds_x=410
mb2_a_s_cap_screw_ds_y=486
mb2_a_s_cap_screw_effect_length=25
mb2_a_s_cap_screw_size=16
mb2_a_s_d_plate_width=600
mb2_a_s_ejp_a_thickness=25
```

Value =

Choose existing expression to edit, or enter new one

Fig. 13 The updated mould base assembly with predefined size "7580"



A	120.0000
B	56.0000
n	2.0000

Fig. 14 "Guide pin pattern" parameter editing UI

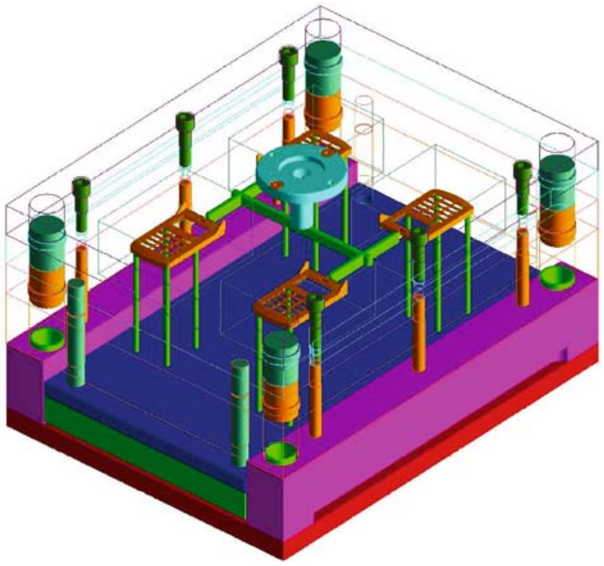


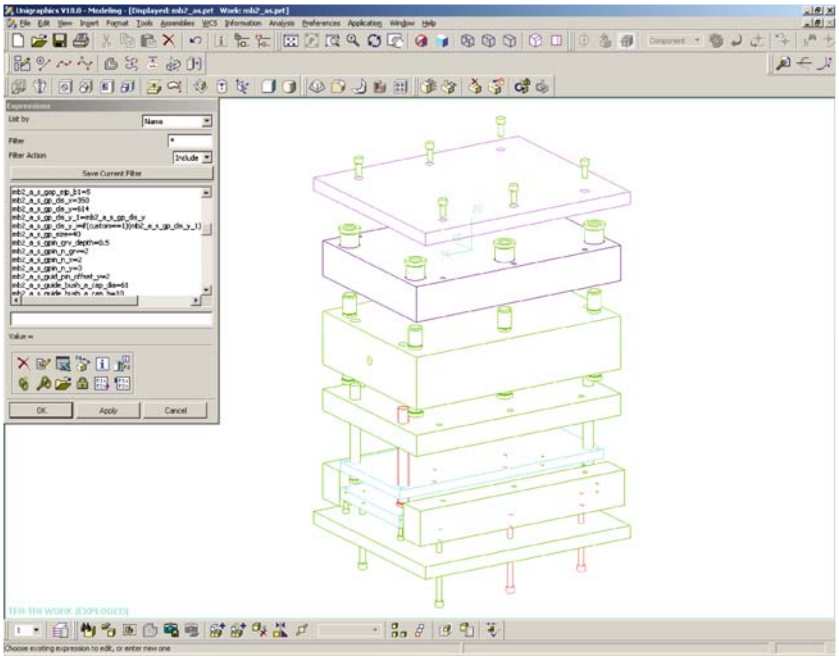
Fig. 16 A mould base assembly (partially loaded)

suitable, he can change it to "2×3" pattern. The UI for editing the pattern is shown in Fig. 14. Assume the number of n is changed to 3 with other appropriate values for A and B (A=540 mm, B=614 mm), then once they are accepted, the assembly can be regenerated automatically as shown in Fig. 15.

Figure 16 shows a portion of a mould base created as part of an actual mould design using QuickMould. In QuickMould, standard components are also managed with a unified design feature approach [10].

7 Summary and conclusion

In this paper, associative assembly features (AAFs) have been modelled in the form of self-contained objects. The object class for the assembly design features has been defined. It is generic in nature and flexible to accommodate different relations among geometrical entities within a product model. With this class definition, associative assembly features can not only be automatically generated but also managed through a set of generic user interfaces. As a theoretical contribution, this paper has given a detailed discussion about associative assembly features in the form of objects. The authors believe that associative feature objects are sufficiently flexible to support deep reasoning and network communication in concurrent and collaborative engineering. Future work is aimed at extending the feature definitions to support design verification.


Unigraphics V18.0 - Modeling - [Displayed: mb2_as.prt Work: mb2_as.prt]
File Edit View Insert Format Tools Assemblies WCS Information Analysis Preferences Application Window Help
Expressions
OK
Apply
Cancel

List by: Name
Filter: *
Filter Action: Include
Save Current Filter

```
mb2_a_s_gap_ep_b1=5
mb2_a_s_gp_ds_x=350
mb2_a_s_gp_ds_y=614
mb2_a_s_gp_ds_y_1=mb2_a_s_gp_ds_y
mb2_a_s_gp_ds_y_=if(custom==1)(mb2_a_s_gp_ds_y_1)
mb2_a_s_gp_size=40
mb2_a_s_gpin_grv_depth=0.5
mb2_a_s_gpin_n_grv=2
mb2_a_s_gpin_n_x=2
mb2_a_s_gpin_n_y=3
mb2_a_s_guid_pin_offset_y=2
mb2_a_s_guide_bush_a_cap_dia=61
mb2_a_s_guide_bush_a_cap_h=10
```

Value =

Choose existing expression to edit, or enter new one

Fig. 15 Regenerated mould base assembly after changing the "guide pin pattern"

Acknowledgements The findings of this paper are based on the implementation of QuickMould, which is a software package for designing plastic injection moulds. It was jointly developed by Nanyang Technological University and Singapore Institute of Manufacturing Technology in a Unigraphics CAD environment. The QuickMould research project was sponsored by the InfoComm Development Authority (iDA) of Singapore.

References

1. Bettig B, Shah J (1999) An object-oriented program shell for integrating CAD software tools. Adv Eng Softw 30(8):529–541
2. Britton GA, Tor SB, Lam YC, Deng Y-M (2001) Modelling functional design information for injection mould design. Int J Prod Res 39(12):2501–2515
3. Brunetti G, Golob B (2000) A feature-based approach towards an integrated product model including conceptual design information. Comput Aided Des 32(14):877–887
4. CASA/SME (2000) Virtual enterprise integration: creating a sustainable manufacturing life cycle. Tech Trend Report. SME, Dearborn, MI. http://www.sme.org/casa. Cited 27 February 2006
5. Deng Y-M, Britton GA, Lam YC, Tor SB, Ma Y-S (2002) Feature-based CAD-CAE integration model for injection-moulded product design. Int J Prod Res 40(15):3737–3750
6. Deng Y-M, Tor SB, Britton GA (2000) Abstracting and exploring functional design information for conceptual mechanical product design. Eng Comput 16:36–52
7. Henson BW, Baxter JE, Juster NP (1993) Assembly representation within a product data framework. ASME Adv Des Autom 65(1):195–205
8. Holland WV, Bronsvoort WF (2000) Assembly features in modeling and planning. Robot Com-Int Manuf 16(4):277–294
9. Ma Y-S, Tong T (2003) Associative feature modeling for concurrent engineering integration. Comput Ind 51(1):51–71
10. Ma Y-S, Tor SB, Britton GA (2003) The development of a standard component library for plastic injection mould design using an object oriented approach. Int J Adv Manuf Technol 22 (9–10):611–618
11. Myung S, Han S (2001) Knowledge-based parametric design of mechanical products based on configuration design method. Expert Syst Appl 21(2):99–107
12. Otto HE (2001) From concepts to consistent object specifications: translation of a domain-oriented feature framework into practice. J Comput Sci Technol 16(3):208–230
13. Qian L, Gero JS (1996) Function-behavior-structure paths and their role in analogy-based design. AIEDAM 10:289–312
14. Shah JJ (1995) Parametric and feature-based CAD/CAM: concepts, techniques, and applications. Wiley, New York
15. Shah JJ, Rogers MT (1993) Assembly modeling as an extension of feature-based design. Res Eng Des 5:218–237
16. Sugimura N, Moriwaki T, Kakino T (1996) A study on assembly model based on STEP and its application to assembly process planning. ASME Japan/USA symposium on flexible automation, Boston, MA, 2:791–794
17. Welch RV, Dixon JR (1992) Representing function, behavior and structure during conceptual design. Proceedings of the ASME design theory and methodology conference, Scottsdale, AZ, 13-16 September, 42:11–18
18. Whitney DE, Mantripragada R, Adams JD, Rhee SJ (1999a) Designing assemblies. Res Eng Des 11(4):229–253
19. Whitney DE, Mantripragada R, Adams JD, Rhee SJ (1999b) Toward a theory for design of kinematically constrained mechanical assembly. Int J Rob Res 18(12):1235–1248
20. Whitney DE, Shukla G, Praun SV (2001) A design procedure applicable to different classes of assemblies. Proceedings of DETC'01, ASME 2001 design engineering technical conferences and computers and information in engineering conference, Pittsburgh, PA
21. Zha XF, Du HJ, Qiu JH (2001) Knowledge-based approach and system for assembly oriented design, part I: the approach. Eng Appl Artif Intell 14(1):61–75
22. Zha XF, Du HJ (2002) A PDES/STEP-based model and system for concurrent integrated design and assembly planning. Comput Aided Des 34(14):1087–1110
23. Zhang WY, Tor SB, Britton GA (2002) A two-level modeling approach to acquire functional design knowledge in engineering systems. Int J Adv Manuf Technol 19(6):454–460