Design, Definition and Implementation of a Component Library for Mechanical Design

Y. -S. Ma, S. B. Tor, Graeme A. Britton and T. Wijaya

***Abstract*--This paper presents the detailed object definition, design, and implementation of a Standard Component Library (SCL), which provides parametric component models for the convenience of designers. With more than 200 types of components from different suppliers being implemented, it is believed that the object design of this module has generic nature and can be expanded for most mechanical standard components in a collaborative environment. The advantages of this implementation are the ease of use and simple customization. In this paper, QuickMould architecture framework is also introduced.**

***Index Terms*—Object Oriented Methods, Manufacturing, Design Automation, and CADCAM**

I. Introduction

PLASTIC injection mould design is a tedious task. Due to market competition, mould designers have to compress their design time as much as possible. To reduce the modification effort, parametric design approach is usually used. Hence, specialized mould design tools based on CAD software are highly demanded. In the market, several mould design packages have been available, e.g. iMouldwork [1], IKMould [2], MoldWizard [3]. Very often, outsourced components (known as standard components) are used. Making 3D parametric models available for such components to mould designers will significantly reduce the design lead-time and cost, and enhance design flexibility. QuickMould is a software package for designing plastic injection moulds in a 3D environment. It is developed on top of Unigraphics CAD / CAM software, and implemented in full object oriented approach [4][5]. This paper focuses on the standard component library (SCL) module of Quickmould. Comparing with other standard component libraries available in the market, QuickMould SCL is very easy to be expanded with user-defined components without programming effort.

In the early half of 1990s, as reported by Johannesson [6], Culley et al. [7] and Lodenstein et al. [8] that, there existed some computer-based standard component catalogues. They contain 2D geometry data and are hard-coded so that the components can be selected and automatically put on the working drawing. Some programs can generate standard components [8] instead of retrieving existing CAD models. However such programs are specifically designed for fixed vendors, and are difficult to be adopted by others due to the variations of component definitions. Qamhiyah [9] presented a strategy for the automatic generation of customized libraries of form features and basic design shapes by coding convexity of B-rep loops and their neighborhoods. However, this method needs the user's input on a theoretical threshold value for convexity coding. To capture more information from designers, Deng et al [10] and Zhang et al [11] extended object representations to mechanical functions and tried to develop a generic scheme for conceptual synthesis domain. Ragli et al. [12] focused on the archival and reuse of design intent by using a 'signature structure'. However, their work requires end-users to have high-level understanding about design engineering background and to cross a major hurdle of adapting to the end-users' knowledge processing cycles. More recently, EDS Inc released Unigraphics v18 software with knowledge fusion (KF) capability [13]. This technology enables the user to capture design intent in a declarative manner in a macro command file. However, such a package requires highly experienced knowledge workers instead of designers. In the recent trend of collaborative engineering, to reduce the dependency to vendors for open-structured geometrical modeling services, Shah et al [14] developed a 3-layer communication architecture to wrap up geometrical modeler functions into a neutral "plug-compatible' definitions. Contradicting to their original objective, to implement their idea, the support from all vendors of geometrical modelers is required. It has been recognized that CAD libraries are very useful for reusing design knowledge and engineering data. Although many software tools have been developed for mould design [1] [2] [15] [16] [17], but it was not well researched on how the end users can incorporate their own components into a predefined library. The research work reported in this paper is aiming to increase the knowledge contents of standard component elements in a CAD library, and to enable users to incorporate new elements with ease in an open architecture.

QuickMould research project was sponsored by former National Computer Board (now renamed as the infocom Development Authority) of Singapore, and jointly developed by Nanyang Technological University (NTU) and Gintic Institute of Manufacturing Technology (Gintic), Singapore. Currently, QuickMould is being used as a research package in NTU.

Y. -S. Ma (telephone: 65-6790-5913, e-mail: mysma@ ntu.edu.sg), S. B. Tor and Graeme A. Britton are associate professors, school of MPE, NTU, 50 Nanyang Avenue, Singapore 639798. S. B. Tor is also the head of manufacturing engineering division.

T. Wijaya is a final year student of NTU.

II. QuickMould Architecture

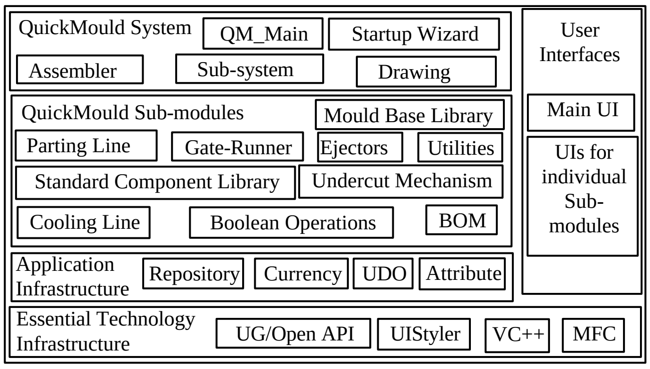


Fig. 1 QuickMould Architecture

Fig. 1 shows the QuickMould 4-layer architecture. The first layer of essential technology infrastructure includes Visual C++ (VC++), Microsoft Foundation Classes (MFC), and API modules provided with Unigraphics software, i.e. UG/Open API, UIStyler. The second layer is the application Infrastructure. This layer contains the basic domain support for QuickMould. "Repository" provides the mechanism to wrap and access QuickMould data in CAD data structure, i.e. UG entities. "Currency" provides the basic classes, e.g. typical assemblies, components, and a table of mould assemblies and components (known as dictionary), that facilitate information representation, sharing and exchanging among various modules/objects in QuickMould. "UDO" provides the mechanism to store QuickMould information through UG's User Defined Objects (UDOs). The key characteristic of UDOs is that they enable persistent links for application objects with UG entities. "Attributes" provides the data repositories of QuickMould attached to appropriate CAD entities. The third layer contains 10 QuickMould functional modules: Parting Line, Undercut Mechanism, Mould Base Library, Cooling Line, Gate-Runner, Standard Component Library (SCL), Ejectors, Bill of Materials (BOM), Global Boolean Operations (GBO) and Global Component Transformation. Each module is a set of specialized design processes chained by user interfaces and implemented with automated or semi-automated QuickMould functions. The top layer of QuickMould architecture contains the system modules, i.e. QM_Main, Start-up Wizard, Assembler, Sub-system and Drawing.

This paper is going to introduce the Standard Component Library (SCL) module only. The functions and design of other modules are out of the scope of this paper. SCL module provides functions such as loading user-selected component templates, modifying dimensions or feature configurations, and updating geometric models. The user can not only select from available components, but also incorporate their own components into the library. QuickMould SCL has contained over 200 types of standard mould components from several manufacturers, such as Misumi, Hasco and DME.

III. Industrial Catelogue

To study existing commercial catalogues, a Misumi catalogue of "Standard Components for plastic Molds" [18] is used. Fig. 2 shows a type of components extracted, i.e. "sprue bushings – shoulder type with taper". Usually, components are classified in a family structure with type identifications. As shown in the figure, this page contains several catalogue numbers, with a form like "SBT_". Here, the '_' at the end of the string reflect the variations in specifications or the feature configurations. For standard components, it is common to have several sets of specifications for a common or a set of geometric configuration(s) in order to meet the needs in different application contexts and to reduce redesign effort. In this "sprue bushings" example, "SBTM", "SBTD", and "SBTS" are different due to materials used while "SBTMH", "SBTDH" and "SBTSH" are the corresponding derived types with string eliminators. These catalogue numbers are referred to as major types. Similarly, under a major type, there could be several feature configurations for the detailed geometry. In Misumi terminology, they are referred to as subtypes and alterations.

In "sprue bushings" example as shown in Fig. 3, the alterations are made for runner openings at the bottom cone, i.e. from "AIW" to "CLQ"; for head geometry ("KC"); for runner holder ("ZC"), and/or for customized length ("LKC"). It can be seen that, although there are different configurations and alterations for a type of component, many common parameters are used. For example in Fig. 2, "D", "L", "SR", "P", "A", are common key parameters. Dimensions are specified according to sizes, and many of them are constrained. Take an example, for the sizes from "10" to "16", the range for the length ("L") is given from 0 mm to 100 mm, while for the sizes above "16", the range can be from 0 mm to 150 mm. As to examples of constrains, some are applicable to all cases, e.g. "D > V >= α+2" while some of them are conditional to a particular configuration or alteration, e.g. "S >= α+2" in the alteration "ZC". Other properties worth mentioning are tolerances, ordering code, delivery lead-time, material specifications, etc. Ideally, all these properties should be included and managed in the proposed CAD component library.

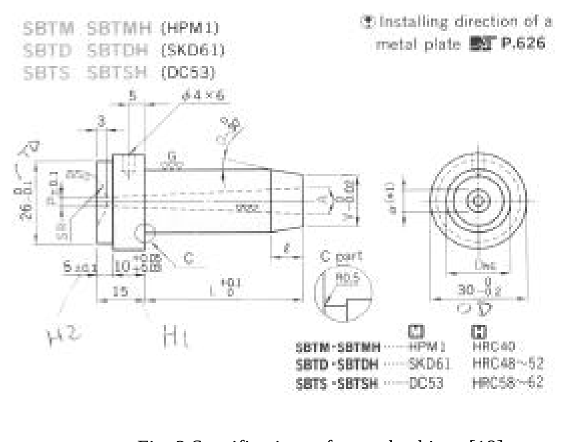
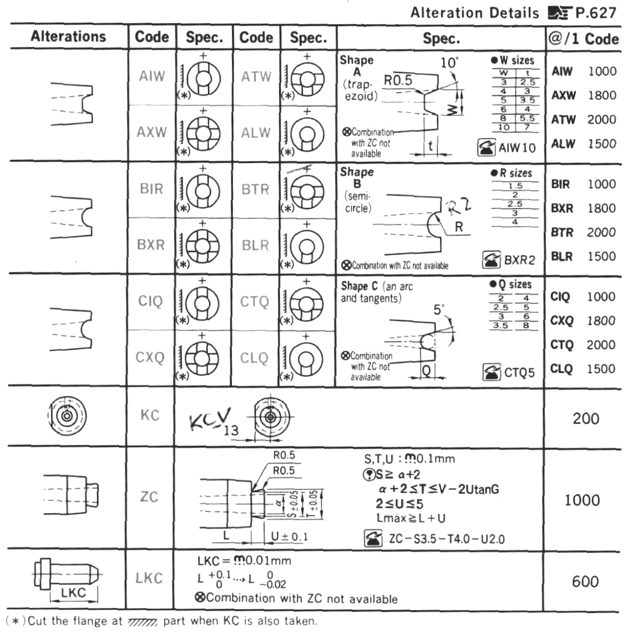


Fig. 2 Specifications of sprue bushings [18]



Alteration Details P.627

Alterations	Code	Spec.	Code	Spec.	Spec.	@/1 Code
	AIW	(*)	ATW	(*)	Shape A (trapezoid) 10° R0.5 W t; ● W sizes W/t: 3/2.5, 4/3, 5/3.5, 6/4, 8/5.5, 10/7; ⊗Combination with ZC not available; AIW10	AIW 1000 AXW 1800 ATW 2000 ALW 1500
	AXW	(*)	ALW	(*)		
	BIR	(*)	BTR	(*)	Shape B (semi-circle) R2 R; ● R sizes 1.5, 2, 2.5, 3, 4; ⊗Combination with ZC not available; BXR2	BIR 1000 BXR 1800 BTR 2000 BLR 1500
	BXR	(*)	BLR	(*)		
	CIQ	(*)	CTQ	(*)	Shape C (an arc and tangents) 5° Q; ● Q sizes 2, 4, 2.5, 5, 3, 6, 3.5, 8; ⊗Combination with ZC not available; CTQ5	CIQ 1000 CXQ 1800 CTQ 2000 CLQ 1500
	CXQ	(*)	CLQ	(*)		
	KC	KCV13				200
	ZC	R0.5 R0.5 α S±0.05 T±0.05 L U±0.1; S,T,U : 0.1mm; S≥ α+2; α+2≤T≤V−2UtanG; 2≤U≤5; Lmax≧L+U; ZC−S3.5−T4.0−U2.0				1000
LKC	LKC	LKC = 0.01mm; $L^{+0.1}_{0} \cdots\rightarrow L^{0}_{-0.02}$; ⊗Combination with ZC not available				600

(∗) Cut the flange at ▨ part when KC is also taken.

Fig. 3 Alterations for sprue bushings [18]

IV. QuickMould SCL Approach

Traditionally, parametric modeling is used to store re-usable CAD models where the geometry is modeled with changeable dimensions. Take Misumi catalogue [18] as an example, it contains 21 categories with ***195*** subtypes. Further more, for each subtype, there could be many alterations. Assuming the average number of alterations for each category is ***8***, then, if we use the traditional parametric modeling approach, there has to be ***1560*** CAD models. This number has already been prohibitive for implementation.

In QuickMould, each type of standard components is implemented with three related files as one item in the library, i.e. a CAD template file, a feature configuration (CFG) file, and a dimension data file. Their relationships are shown in Figure 4. In the run time, the variations of feature configurations, alterations, dimensions, constraints, etc. are captured as associated properties in an object.

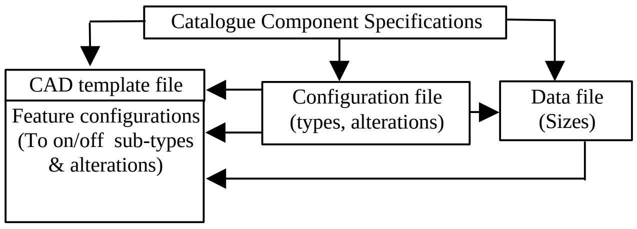

Catalogue Component Specifications
CAD template file
Feature configurations (To on/off sub-types & alterations)
Configuration file (types, alterations)
Data file (Sizes)

Fig. 4 Conceptual relationships among the three files

Based on existing component catalogues, standard components are classified systematically according to their categories (or major types) and features, alterations, dimensions, etc. With a formal convention, each component structure can be consistently described in an ASCII text file. It is called feature CFG file in QuickMould terminology. Such a file specifies the choices for subtypes, alterations, and key parameters used. The contents of CFG files will be further introduced in sub-section B. A parametric CAD template has to be developed to model the geometry. A dimension data file is used to store predefined dimension values provided by the vendors as the default for different sizes.

In such an approach, the library program can load the CAD template model into the working assembly design, and then based on the information read from the feature configuration and dimension data files, the user can select subtypes, alterations, and dimension values (Fig. 5(a)), etc. The user can also further modify the detailed dimensions to meet his exact requirements (Fig. 5(b)).

A. *CAD Model Templates*

Each major type of components is represented by one template. Hence a standard component template is loaded according to the selection of the major type. In each pre-defined template, geometrical features are parametrically defined, and those features related to subtypes and alterations are embedded in the template as optional ones. They are controlled by "logical expressions" [3], which can suspend or activate features according to feature configuration requirements.

In the template file, there could be two solids, i.e. the real component solid (compulsory), and a "space" solid (optional), which is defined as the typical space required for accommodating the real component in assembly. They are defined in a parametric manner and associative to each other. Through "space" definitions, the related features on other components can be made associated to the standard components used. Hence, it enables associative modifications with the native UG reference mechanism, including positional and dimensional changes.

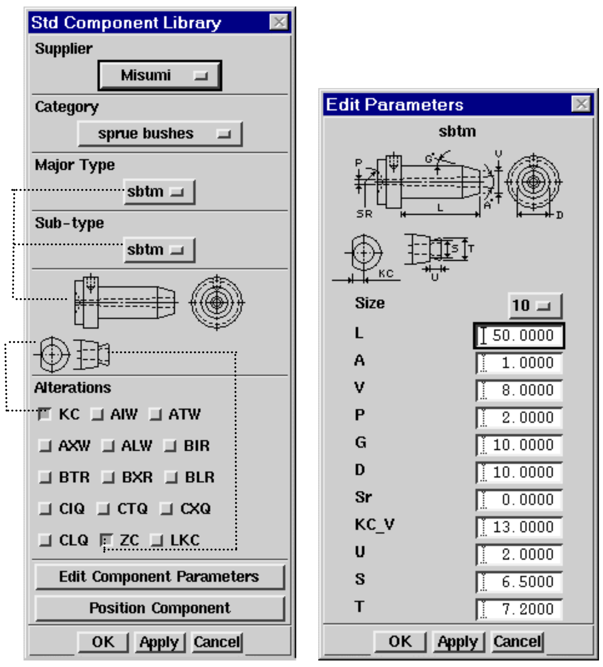
Geometric expressions are grouped into two levels. The first level contains those that are related to the key component parameters, and are essential for the function of the component or related to other components in assemblies. The second level expressions are for component geometry constructions, where detailed feature dimensions are defined. This level of expressions is made related to the key expressions. When updating those key expressions by program, the lower level expressions are subsequently updated automatically. CAD model template files are organized in a directory tree structure according to the suppliers, categories and major types.

B. *Feature Configuration (CFG) Files*

Standard component library is designed to be generic to all types of mechanical parts. To enable this requirement, feature CFG files are used to define individual components with a standard convention.

A feature CFG file (e.g. sprue_bushes.cfg for the example shown in Fig. 2 and 3) is started with the catalogue numbers ("sbtm, sbtd, … sbtsh"), which are named as "major types". In this example, there are six major types specified. Next, the related CAD template and the dimension data filenames are given, e.g. "sbt_.prt" and "sprue_bushes.dat". The following record specifies the parameter symbol representing the size. In this case, “D” is specified. Then, in the next section, the available subtypes are listed. Among subtypes, they are exclusive to each other. This means that at any time, only one of them is effective. Subtype names read from this CFG file will also appear on the user interfaces for the user’s selection when loading the component (see Fig. 5(a)). They provide the choices available for subtypes and the keys to change the effective configuration for the loaded component.

Next, alterations are organized as groups. Similar to the subtypes, in each group, member alterations are exclusive to each other. Their specific properties are separately grouped. For the “sprue bushings”, alterations are divided into three groups (see Fig. 3). Alteration “KC” alone is one group while alterations “AIW” to “ZC” belong to the second group. Alteration “LKC” forms the third one. It is clear that combinations of alterations from the same group are not meaningful. To enforce this rule, on the user interface as shown in Fig. 5, only one alteration can be toggled on for each group. For example, combination of “AIW” and “ZC” is not available as given in the catalogue (Fig. 3). For alterations from different groups, they can coexist simultaneously. For example, “KC” and “ZC” alterations can be selected together.


Std Component Library

Supplier: Misumi

Category: sprue bushes

Major Type: sbtm

Sub-type: sbtm

Alterations: [✓ KC] AIW ATW AXW ALW BIR BTR BXR BLR CIQ CTQ CXQ CLQ [✓ ZC] LKC

Edit Parameters

sbtm

Size	10
L	50.0000
A	1.0000
V	8.0000
P	2.0000
G	10.0000
D	10.0000
Sr	0.0000
KC_V	13.0000
U	2.0000
S	6.5000
T	7.2000

(a) SCL main UI (b) Parameters editing UI

Fig. 5

The next record type is for key parameters and their corresponding expressions in the CAD template, where they are associated to the geometrical features of the CAD model. SCL distinguishes two types of key parameters, those to be displayed on the loading/editing user interfaces and those that are not supposed to be modified/customized by the user. In the given example, key parameters “L”, “A”, … “SR”, etc. are displayable while “OD” and “α” are not (see Fig. 2 and Fig. 5(b)). All parameters are assigned with the values stored in the dimension data file when the component is initially loaded, but the user can only modify displayable ones through SCL UIs.

For a component, constraints are also included in the feature CFG file. The format likes “V >= α+2”. They are very important to keep features valid and data consistent. These constraints are read into the component object buffer and being checked whenever necessary to make sure they are satisfied.

In the CFG file, sections are arranged according to its applicable scope. So far, what have been introduced are the attributes or properties that are applicable to all subtypes and alterations. There are attributes defined in specific sections that are unique to a particular major type, subtype, or an alteration, e.g. icon bitmap files for subtype/alteration identification, some key dimensions, constraints, etc.

Tolerances for each parameter are recorded with the upper and lower limits. Since they are associated with key parameters, and they can be updated and listed according to the application. This provides a possible link to the technical specifications when creating the component/mould plate drawings. Other additional attributes, such as material, default delivery time, prices, etc. are included as well. For example, the purchasing order has to give specific code for the item name, type, size, alterations etc. A method has been implemented in SCL class to interpret the format given in the CFG file, e.g. “sub_type @D-@L-SR-P-A-V-G-alterations-end”, and then retrieve the values from the relevant key parameters from CAD model and generate the order code as “SBTM 25 – 45.5 – SR23 – P4.5 – A4 – V20.0 – G5-AXW8-KC”. In fact, the prices for certain companies, like MISUMI, is also set according to the type, sizes and dimensions of the components, hence, it has to be derived automatically from the CAD model parameters.

Different companies may use quite different format of catalogues and hence the major types, sub types, etc need to be organized differently. However, the approach to organize CAD libraries and the generic data (attributes) structure is still valid due to the commonality of industrial practices.

C. *Dimensional Data Files*

For each size of a standard component, the vendor usually provides a set of default parameter values so that its manufacturing activities can be predefined and cost minimized. The dimension data file contains the default values of key parameters for all the available sizes. Each parameter is considered to have a range from the minimum to the maximum with a minimum increment except its current value. Allowed ranges specified are checked when the user edits parameters with the UI shown in Fig. 5(b).

V. Data Structure of “QM_STD_COMP” Class

To represent a standard component in a CAD session, a class named as “QM_STD_COMP” has been defined. To simplify the presentation of the class definition structure, a

“Macro” type pseudo-program format is used in the following description below. For simplicity reason, a single property or function mentioned in this paper could represent a single or a cluster of attribute(s) or method(s).

A “QM_STD_COMP” object’s properties can be clustered into two groups, i.e. persistent properties and buffer ones. Persistent properties are static and stored in the form of CAD attributes. They are associated to a CAD component pointer during the run-time session. The component pointer is included in the “QM_STD_COMP” object properties. However, similar to most of CAD systems, UG initiates entities during the run time as a data structure, and the pointer to an entity is assigned during the run time and is only valid for the current session. On the other hand, a “QM_STD_COMP” object has to be associated with a CAD component persistently such that its private properties are stored after closing the current UG session and can be retrieved when QuickMould reinitiates the object in the next session. This problem is solved by using User Defined Objects (UDOs).

QuickMould maintains a list of its entity pointers named as currency dictionary. With this dictionary, QuickMould objects are dynamically mapped with their corresponding UG geometric entities. Among persistent properties, the name of supplier, category and major types provide the input to locate the feature CFG file with a predefined naming convention for library directories and files. Once the CFG file is identified, the CAD template and the dimension data files can be uniquely retrieved.

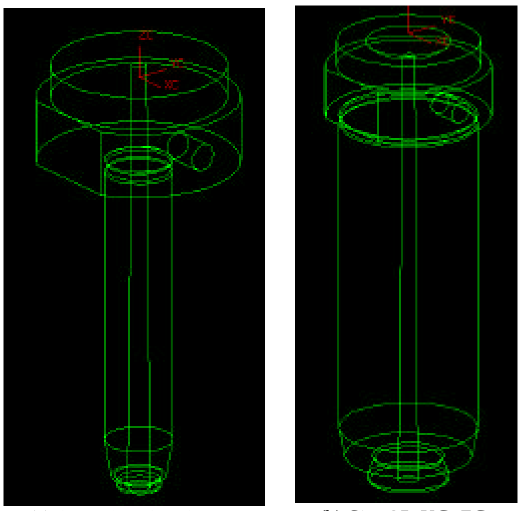

(a) Size 10, KC, ZC (b) Size 25, KC, ZC

Fig. 6 A sprue bushing component loaded

The properties contained in the buffer block are for editing and supporting UI functions because typically, SCL allows user to set their choices and then confirm by clicking “Apply” or “OK” button at the bottom of the UI (See Fig. 5). In another situation, when the user edits an existing “QM_STD_Comp” object, they may cancel their modifications made with UI elements. Others include alteration groups that allows the user to select, constraints to verified before accepting the user’s input, displayed parameters that are editable by the user. Here, displayed parameters have to be differentiated from key parameters. Note that some properties are associated to the CFG file. For example, the effective constraints, key parameters and their expressions, and displayed parameters have to be updated systematically for different combinations of major types, subtypes, alterations, sizes, etc.

As to the class methods, there are four levels. The most bottom level is the access methods. The second level methods are common “foundation functions”, including string manipulation, expression interpretation and evaluation, etc. The third level is “functional methods” to cater for parameter editing, e.g. expressions backing-up, evaluation, modification, updating and restoring. Functions in this level also include “Activate_sub_type()”, “Generate_size ()”, etc. These methods are enablers that provide the flexibility for a predefined standard component to be applied with different configurations and dimensions. The highest level is “Application Methods”. Their functions are elaborated in the following section in more detail.

VI. QuickMould SCL Applications

A. *Adding a Mould Component*

When the user starts the SCL module, a “QM_STD_COMP” object is first initiated. SCL module will assign the object attributes step by step. To add a component, the user needs to select a parent assembly part to which the component is added. Then the SCL module main dialogue box (Fig. 5(a) is generated. The program checks the available catalogues in different directories and creates a pull-down menu under the “Supplier” button. Once the user confirmed the supplier, e.g. “Misumi”, then the defined component templates are checked in this catalogue. They are listed in two levels, i.e. “Category” and “Major Type” in the UI as shown in Fig. 5(a). Other available choices, such as subtypes and alterations, are retrieved from the feature CFG file, and incorporated into the UI. The user can select different items from the pull-down menus. Note that the bitmaps indicating the active selections are displayed in the UI. For adding a new component, the user also needs to click “Edit Component Parameter” button, SCL will then display the “Edit Parameters” dialogue box, as shown in Fig. 5(b). With this UI, the user can select the size he wants. All the default parameters related to each size are retrieved from the dimension data file. Finally, the user clicks “Apply” button to load the selected component.

At this moment, appropriate “QM_STD_Comp” object attributes are instantiated. It calls class methods to complete the following steps: (a) insert the loaded component into assembly, (b) retrieve key dimensions from the dimension data file and update the geometry; (c) rename the component to localize it in the current project directory; (d) convert the component into a QM component and list it in the currency dictionary. Then the loaded component is shown as in Fig. 6(a). If the user wants to modify the size, the user can click on the “size” button shown in Fig. 5(b), e.g. from “10” to “25”, then the key parameters are updated. Once the user clicks "OK" or "Apply", the CAD model are updated accordingly (see Fig. 6). When the QM exits from the SCL scope, this "QM_STD_Comp" object's properties are mapped back to the attributes of the component, and this "QM_STD_Comp" object is destroyed.

B. Modifying an Existing Mould Component

To edit an inserted standard component, the SCL module runs a method to let the user select the target, then all the QuickMould related attributes associated with the selected component are retrieved and the corresponding attributes in this "QM_STD_COMP" object will be updated. The same SCL UI as shown in Fig. 5(a) is then established after searching available library items as well as the attributes of the existing component. Actually, its feature CFG file is read again, and the information for the available configurations and the respective selection option menus and buttons are updated. This standard component can now be edited. The rest procedures are very similar to the adding process; hence they are not further described.

C. Deleting a standard component.

Methods have been implemented to delete the selected standard component from the project assembly, and delete its part, configuration and data files from the current project directory as well. Simultaneously, the related currency object and dictionary pointer in the session are also cleaned.

VII. Discussion, Conclusions and Suggestions

Considering the possible big number of combinations of major types, subtypes and alterations, if the implementation were done for every type, the implementation could be tedious. However, after the implementation, it has been demonstrated that this approach can be very efficient. This is because CAD templates, configuration and data files can be combined/shared as much as possible. The big advantage of this approach is that the end users do not need to program the library elements but just to follow certain procedural conventions to produce the template models and a pair of text files. For future work, it is suggested to automate the configuration combinations based on some common elements of features with generative approach. Considering the scalability for an Application Service Provider (ASP) business model, due to large amount of data to be managed, the authors suggest an XML based Object-oriented database to be built to support the library. Further research work is still needed.

In summary, this proposed CAD library is flexible enough for a wide range of standard components. They are defined by using component templates, CFG files and data files. The reported data structure and software module framework can be used as a foundation for future collaborative design environment over the Internet.

References

[1] M. W. Fu, J. Y. H. Fuh, and A. Y. C. Nee, "Core and Cavity Generation Method in Injection Mould Design," Int. J. of Prod. Res. Vol. 39, pp. 121-138, 2001.

[2] C. K. Mok, K. S. Chin, and John K. L. Ho, "An interactive knowledge-based CAD system for mould design in injection moulding processes," *Int. J. of Advanced Manuf. Technology* vol. 17: pp. 27-38, 2001.

[3] Unigraphics Solutions Inc. UG Documentation Help. Maryland Heights, MO, 2000.

[4] G. A. Britton, Y. -S. Ma and S. B. Tor, "Object Technology Development and Unigraphics", Proceedings of Unigraphics User Group 1999 Spring Conference: Manage design evolution, Newport Beach, California, USA.

[5] G. Britton, S. B. Tor and Y. Wang, "Virtual concurrent product development of plastic injection mould", Proc Instn Mech Engrs Vol. 214 Part B, pp. 165-168, 2000.

[6] H. L. Johannesson, "Computer Aided Part Design Based On Standard Component Interface Geometry, " Advances in Design Automation, vol. 32-2, pp. 347-352, 1991.

[7] S. J. Culley and S. J. Webber, "Implementation requirements for electronic standard component catalogues," Proceedings of the IME Part B: J. of Eng. Manuf., vol. 206, pp. 253-260, 1992.

[8] M. A. Lodenstein, D. M. Romps, and P. Tran, "Development of mold design software," Paper presented at the Society of Plastic Engineers Annual Technical Conference – ANTEC, Brookfield, CT 1, pp. 1090-1093, 1994.

[9] A. Z. Qamhiyah, "A Strategy for the Construction of Customized Design Libraries for CAD," Computer Aided Design, vol. 30, pp. 897-904, 1998.

[10] Deng Y. –M., Britton G. A., and Tor S. B., "A design Perspective of Mechanical Function and Its Object-Oriented Representation Scheme", Engineering with Computers, V14, n4, 1998, pp. 309-320.

[11] W. Y. Zhang, S. B. Tor, and G. A. Britton, "A prototype knowledge-based system for conceptual synthesis of design process", Int. J. Adv. Manuf. Tech. vol.17, pp. 549-557, 2001.

[12] W. C. Regli, V. A. Cicirello, "Managing Digital Libraries for Computer-Aided Design," Computer-Aided Design, vol. 32, pp. 119-132, 2000.

[13] EDS Inc, "Knowledge Fusion for Designers", Student Guide, MT15130 – Version 18.0, November 2001.

[14] J. J. Shah, Hiren Dedhia, Viren Pherwani and Sachin Solkhan. "Dynamic interfacing of applications to geometric modeling services via modeler neutral protocol", Computer-Aided Design, Vol. 29, No 12, pp 811-824, 1997.

[15] S. W. Lye, and H. Y. Yeong, "Computer-Assisted Mould Design for Styrofoam Products," Computers in Industry, vol. 18, pp. 117-126, 1992.

[16] S. B. Tor, S. G. Lee, Y. H. S. H. Chung, "A two-stage collapsible core for injection moulded plastic parts with internal undercuts", Int. J. of Machine Tools and Manufacturing, Vol. 40, pp1215-1233, 2000.

[17] T. L. Neo, K. S. Lee, "Three-Dimensional Kernel Development for Injection Mould Design," Int. J. of Advanced Manuf. Tech., vol. 17, pp. 453-461, 2001.

[18] MISUMI Corporation, Face Standard Components for Plastic Molds (May 1993 -> Apr. 1996), 2-CHOME, TOYO, KOTO-KU, TOKYO, 135, JAPAN.