Int J Adv Manuf Technol (2003) 22: 611–618
DOI 10.1007/s00170-003-1555-8

ORIGINAL ARTICLE

Yong-Sheng Ma · Shu Beng Tor · Graeme A. Britton

The development of a standard component library for plastic injection mould design using an object-oriented approach

Received: 31 July 2002 / Accepted: 7 October 2002 / Published online: 2 August 2003

Abstract Very often, outsourced components (known as standard components) are used for reducing design effort and product cost in mould design. They are usually manufactured by specialised suppliers. 3D CAD parametric models of these components will significantly reduce mould design lead-time and cost, and enhance design flexibility. This paper presents the detailed object definition, design, and implementation of a Standard Component Library within a mould design software package, QuickMould. With many components from different suppliers implemented, it is believed that the object design has generic nature and can be expanded to include most mechanical components in a collaborative environment. The advantages of this implementation are the compressed data structure, the ease of use and simple customisation.

Keywords Standard components · CAD · Design automation · Collaborative engineering · Object-Oriented · Plastic injection mould design

1 Introduction

CAD libraries are very useful for reusing design knowledge and engineering data. Computer-based standard component libraries with interfaces to CAD systems have been available since the early 1990's [1, 2, 3, 4]. Early libraries contain only 2D or wire frame geometry, and cannot be directly used in 3D parametric design. Recently, several mould design packages and commercial part libraries are available [5, 6, 7, 8, 9, 10, 11]. Some of them provide predefined 3D components. Of these, most of them generate or load in components with predefined features and dimensions in a hard-coding approach. They are either very limited in editing functions, which is difficult to adapt them for different vendors due to the variation in component definitions; or simple CAD files that are numerous and not easily reusable by users.

Some researchers have tried to overcome these problems through geometry classification and capturing design intent [12, 13, 14, 15, 16]. Unfortunately, to apply their approaches, end-users have to have high-level understanding about engineering design, and to adapt their practices to the required knowledge representation schema. More recently, EDS Inc released knowledge fusion capability [9], which can be used to generate new designs with design templates in a declarative manner. The development of these templates requires a highly experienced CAD user who is familiar with software object concepts.

Another issue is to ensure that mould makers use the same set of data consistently for different applications, such as Conceptual design [17, 18], engineering analysis with CAE tools [18,19,20,21] and process planning [12, 22, 23]. Technology development in these areas has been slow, partly because the component representations in CAD files do not include rich non-geometrical information for the related purposes.

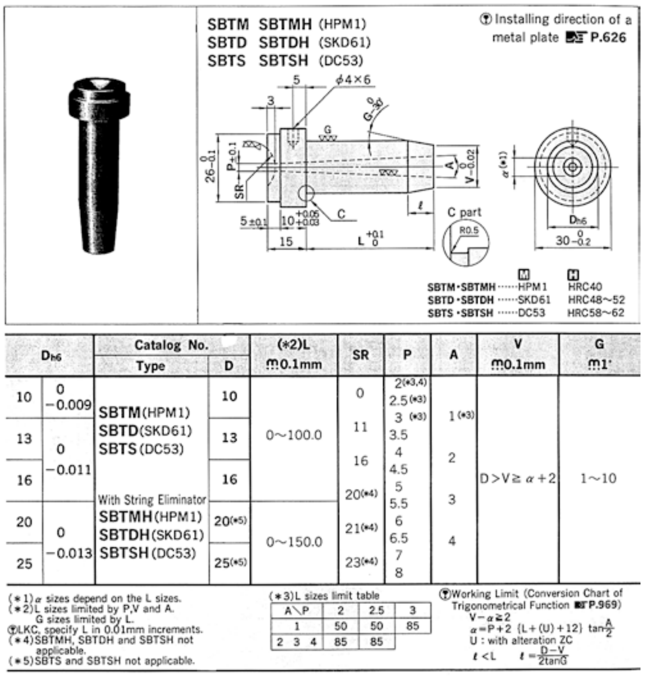
From the review so far, it can be seen that currently available standard component libraries are not satisfactory and new research work is still needed. This paper presents an Object Oriented design of a standard component library (SCL) for mould design. It is believed to be flexible to deal with both dimensional and topological variations, and can be easily extended to other types of mechanical components.

Y-S. Ma (✉) · S. B. Tor · G. A. Britton
Design Research Center, School of Mechanical and Production Engineering,
Nanyang Technological University,
Nanyang Avenue, 639798 Singapore, Singapore
E-mail: mysma@ntu.edu.sg

2 Current industrial practice

Misumi is one of the most popular catalogues for mould design [24]. Components are classified in a family structure with categories, major types and subtypes. As illustrated in Fig. 1, in "sprue bushings" category, several major types in the form like "SBT_" share the same geometry definition. "SBTM", "SBTD", and "SBTS" are different due to materials used, while "SBTMH", "SBTDH" and "SBTSH" are the corresponding derived types with a string eliminator. Similarly, under a major type, there could be several feature configurations known as subtypes. Another type, "precision stepped center pins", is shown in Fig. 2. In this case, the major type is "CPVB-5_". "CPVB-5A" to "CPVB-5E" are five subtypes. In addition to subtypes, many alterations are used for the detailed geometry. In Fig. 1b, the alterations are made for runner openings at the bottom cone, ("AIW"... "CLQ"), head geometry, ("KC" and "ZC"), and/or customized length ("LKC").

It can be seen from Fig. 1a that there are many common parameters, e.g. "D", "L", "SR", "P", "A", etc. Dimensions are specified according to sizes and they may be constrained. In Fig. 1b, the range for the length ("L") is 0–100 mm for the sizes from "10" to "16", and 0–150 mm for those sizes greater than "16". Some constraints are applicable to all cases, e.g. "D > V > = α + 2", while others are particular to subtypes or alterations, e.g. "S > = α + 2" in the alteration "ZC". Other properties include, but not restricted to, tolerances, ordering code, delivery lead-time, and material specifications.


SBTM SBTMH (HPM1)
SBTD SBTDH (SKD61)
SBTS SBTSH (DC53)
Installing direction of a metal plate P.626
5
ϕ4×6
3
G
P±0.1
26−0.1
SR
A
V−0.02
α(*1)
5±0.1
10+0.05 +0.03
C
ℓ
C part
R0.5
15
L +0.1 0
Dh6
30−0.2

SBTM·SBTMH ······HPM1 HRC40
SBTD·SBTDH ······SKD61 HRC48~52
SBTS·SBTSH ······DC53 HRC58~62

Dh6		Catalog No. Type	D	(*2)L ⅿ0.1mm	SR	P	A	V ⅿ0.1mm	G ⅿ1°
10	0 −0.009	SBTM(HPM1) SBTD(SKD61) SBTS(DC53)	10	0~100.0	0, 11, 16, 20(*4), 21(*4), 23(*4)	2(*3,4), 2.5(*3), 3 (*3), 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 8	1(*3), 2, 3, 4	D>V≧α+2	1~10
13	0 −0.011		13						
16			16						
20	0 −0.013	With String Eliminator SBTMH(HPM1) SBTDH(SKD61) SBTSH(DC53)	20(*5)	0~150.0					
25			25(*5)						

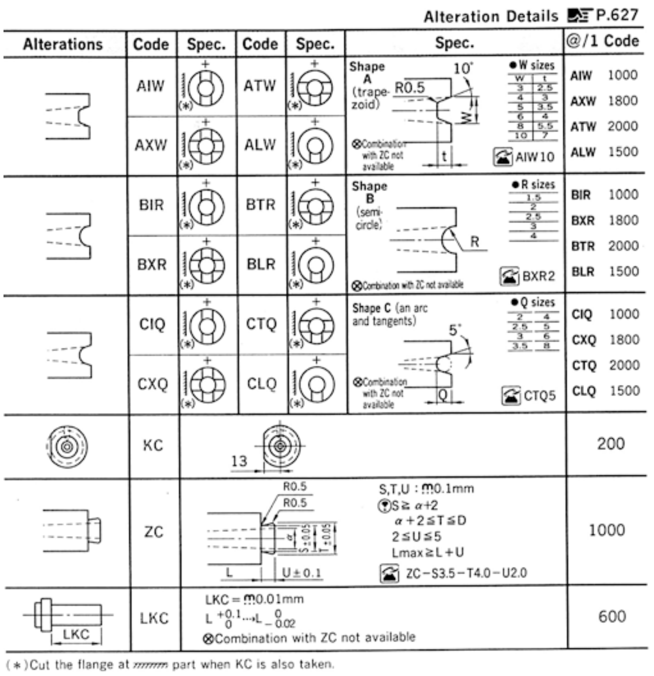
(∗1)α sizes depend on the L sizes.
(∗2)L sizes limited by P,V and A. G sizes limited by L.
LKC, specify L in 0.01mm increments.
(∗4)SBTMH, SBTDH and SBTSH not applicable.
(∗5)SBTS and SBTSH not applicable.

(∗3)L sizes limit table

A\P	2	2.5	3
1	50	50	85
2 3 4	85	85	

Working Limit (Conversion Chart of Trigonometrical Function P.969)
$V-\alpha \geqq 2$
$\alpha = P + 2\{L + (U) + 12\}\tan\frac{A}{2}$
U : with alteration ZC
$\ell < L \quad \ell = \frac{D-V}{2\tan G}$

(a)


Alteration Details P.627

Alterations	Code	Spec.	Code	Spec.	Spec.	@/1 Code
	AIW	(*)	ATW	(*)	Shape A (trape-zoid) R0.5 10° W t ⊗Combination with ZC not available. ●W sizes: W 3, 4, 5, 6, 8, 10; t 2.5, 3, 3.5, 4, 5.5, 7. AIW10	AIW 1000 AXW 1800 ATW 2000 ALW 1500
	AXW	(*)	ALW	(*)		
	BIR	(*)	BTR	(*)	Shape B (semi-circle) 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, 2.5, 3, 3.5; 4, 5, 6, 8. ⊗Combination with ZC not available. CTQ5	CIQ 1000 CXQ 1800 CTQ 2000 CLQ 1500
	CXQ	(*)	CLQ	(*)		
	KC	13				200
	ZC	R0.5 R0.5 α S±0.05 T±0.05 L U±0.1 S,T,U : ⅿ0.1mm S≧α+2 α+2≦T≦D 2≦U≦5 Lmax≧L+U ZC−S3.5−T4.0−U2.0				1000
LKC	LKC	LKC = ⅿ0.01mm $L^{+0.1}_{0} \cdots L^{0}_{-0.02}$ ⊗Combination with ZC not available				600

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

(b)

Fig. 1 "Sprue bushings" standard component type [24]

3 QuickMould standard component library

QuickMould is an application software package for designing plastic injection moulds in a 3D environment that is fully integrated with Unigraphics (UG) CAD software. It was developed using an object-oriented approach [25, 26]. A Standard Component Library (SCL) module is provided to allow designers to select, load, identify and edit standard components. The component representation includes rich information such as suppliers, major types, sub-types, alterations, sizes, constraints, tolerances, etc. Such information can be encapsulated, retrieved and processed for all the library elements generically in object format. More importantly, in this SCL module, each *category* of component is implemented as one item in the library. The variation of feature configurations, dimensions, constraints, and other related information are embedded in it with a standard convention. For example, "sprue bushings" is one item and "precision stepped centre pins" is another. The parametric definition of components allows for variation in topology, as well as dimensions. This is implemented using three related files, a CAD template file, a feature configuration file, and a dimension data file (described below). The files are organized in a directory tree structure according to suppliers and categories.

4 CAD template file

Each category is represented by one CAD template file. In each template, geometrical features are parametrically defined using expressions. Furthermore, the features related to subtypes and alterations are grouped and embedded as optional features. They are controlled by "logical expressions" [9], which can suspend or enable features as required. That is, by manipulating the controlling expressions of the template, major types, subtypes or alterations can be activated or suppressed.

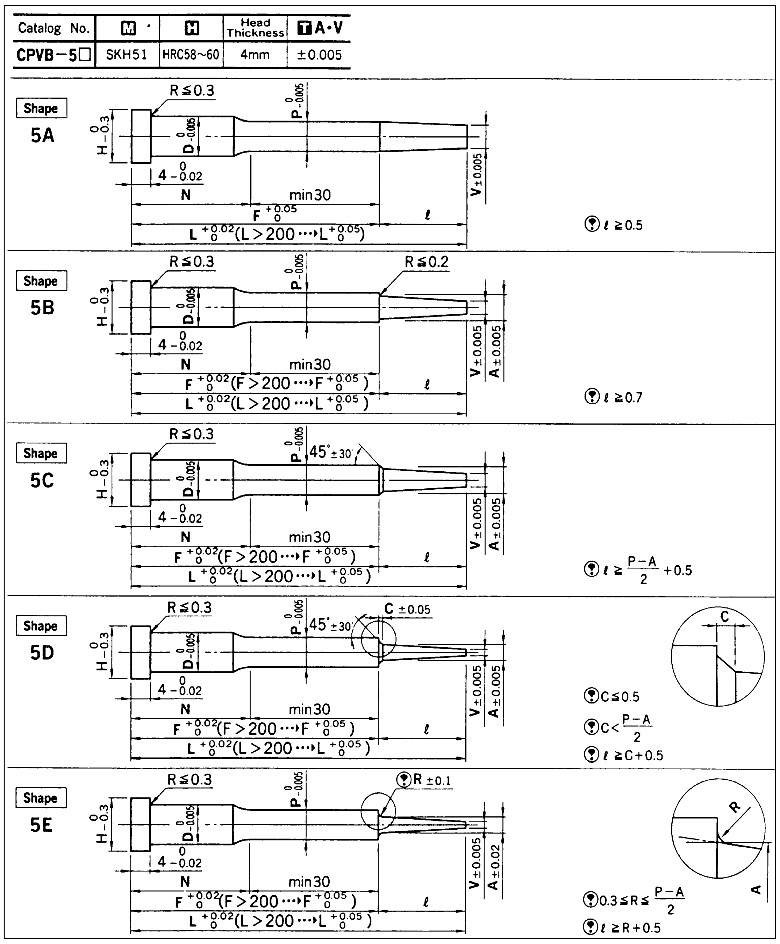


Fig. 2 Subtypes of precision stepped centre pins [24]

Other expressions are grouped into two levels. The first level contains those related to the key component parameters, which are essential for the function of the component or related to other components in the mould assembly; they are named as key expressions. The second level expressions are for component geometry constructions, where detailed feature dimensions are defined; they are referred to key expressions to ensure being updated automatically if the first level expressions are changed.

5 Feature configuration file

A feature configuration file is used to define the structural contents and options of a component category using a standard convention. The structure of the file is generic and hence can be used to define many different types of mechanical components. The file specifies the following: applicable major types, subtypes and alterations; CAD template, dimensional data and bitmap files; size; key parameters; parameter constraints; global parameters that can be edited and the corresponding expressions in the CAD template; additional attributes such as material, default delivery time, prices; etc. The configuration file records are coded to indicate the record interpretation format, which has been predefined and recognized by the record reading methods. Feature configurations are grouped in such a way that members in a group are exclusive to each other while those from different groups can coexist. For example, in Misumi catalogue, at any time, only one subtype is effective. By using UG feature control mechanism, the features related to these configurations are activated or suspended accordingly. Therefore it is important for the CAD template to be modelled in a disciplinary manner such that each feature configuration is fully and independently controlled by a control expression. A similar mechanism is applied to deal with alterations.

The next section contains key parameters, and their corresponding expression names, which are associated to the geometrical features of the CAD model. Two types of key parameters are distinguished: those to be displayed and edited, and those that are hidden and non-editable. Constraints are also included in the format, e.g. "A≥B", and they are read into the component object buffer and are checked in the appropriate context to ensure they are satisfied.

Certain attributes are unique to specific configurations of major types/subtypes/alterations. These properties can be the icon bitmap files for subtype/alteration identification, key dimensions, constraints, etc. They are specified with applicable scopes.

Other general information can be included in the configuration file, for example, tolerance and purchase information. Each tolerance element is represented by the parameter, its upper and lower limits. The limits can be expressions that refer to other parameter values. As such, tolerances are made associated with key parameters, and they can be updated and listed according to the application. This provides a possible link to the component / mould plate drawings. In future, tolerances [12, 22, 23] can also be used by Computer-Aided Process Planning (CAPP) systems. For purchasing applications, ordering information can be given in the format required by vendors, like "sub_type @size-@KeyParameterList-alterations-end". A method has been implemented to generate the effective order code by interpreting parameters and then retrieve their values from CAD models. Hence, it enables automatic updating for purchasing documents or databases.

6 Dimensional data file

A dimension data file is used to store predefined dimension values provided by the vendors as default, organized according to sizes. In the SCL data structure, each parameter is represented by four variables, current, minimum, incremental, and maximum values. The current values of parameters are used to update the CAD expression values when the user selects a particular size, and then the CAD model is updated in turn according to the selected size. Other values that specify the allowable range are very useful for the SCL parameter editing UI functions.

7 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. A "QM_STD_COMP" object's properties are clustered into two groups, the persistent properties and a buffer block. Persistent properties are associated with a CAD component pointer during run time session via a mapping link when the "QM_STD_COMP" object is active. They are static and stored in the form of CAD attributes. This is not a simple task to achieve. On the one hand the CAD 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 model such that its properties are persistent after closing the current UG session and can be retrievable in the next session. To solve this problem, UG's User Defined Objects (UDOs) are used. They can keep links connected to CAD entities persistently. QuickMould maintains a list of earmarked UDO pointers known as the "currency dictionary". Each pointer is associated to the geometric entity via an UDO. When QuickMould is initiated, it searches all QuickMould UDOs and establishes the currency dictionary. Therefore QuickMould objects are dynamically identified and mapped with their corresponding CAD geometric entities.

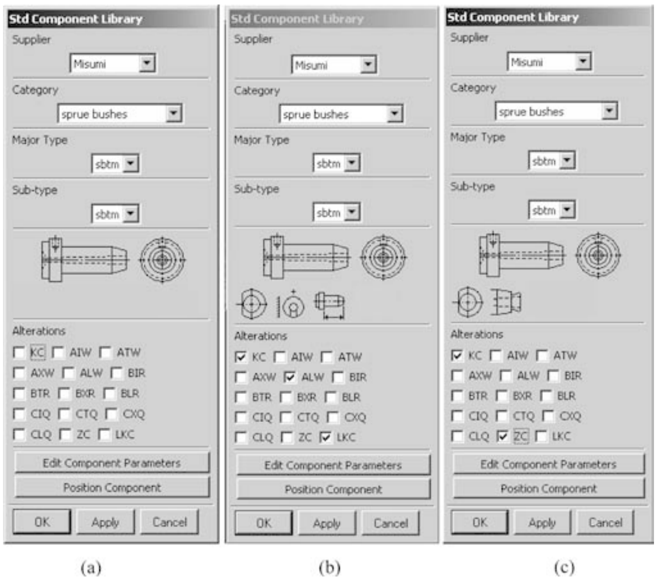
The persistent properties like the supplier name, category and major types, provide the input to locate the feature configuration file using a predefined naming convention for library directories and files. Once the configuration file is identified, the CAD template file as well as the dimension data file of each QM_STD_COMP object can be uniquely retrieved. Typically, SCL allows user to set their choices and then confirm them. To facilitate cancellation of modifications to components, a buffer is used. This buffer supports editing and UI functions. It contains the modified attributes that are to be eventually saved to the UG repository as "permanent" attributes, or to be deleted. Other than the mapped copies of persistent properties, it also contains alteration groups that allow the user to select, constraints to be verified before accepting the user's input, and displayed parameters that are editable by the user.

Class methods for QM_STD_COMP class have been grouped into four levels. The lowest level contains the access methods implemented to retrieve and set all properties of the class objects. The second level methods are for common operations required to support object functions including string manipulation, expression interpretation and evaluation, etc. The third level is named as "functional methods" to cater for parameter editing, backing up expressions, restoring for "undo" purpose, and switching among feature configurations. To support UI editing fields, methods to evaluate current parameter values, ranges, constraints, and even the order code are also implemented at this level. The highest level of this class's methods is called "Application Methods", which are introduced as key scenarios in the next section.

8 Standard component library scenarios

8.1 Adding a standard component

First, a blank "QM_STD_COMP" object is initiated. SCL module will assign the object attributes step by step. The module searches the library directory, identifies all library items and groups them into a "supplier-category-major type" tree structure. The main UI (see Fig. 3a) is initiated. Once the user makes choices from the available suppliers and categories, immediately, the feature configuration file is identified and interpreted. The information from the file defines the properties of the component object such as the CAD template filename, its major type name, available subtypes and alterations and their corresponding bitmap filenames and toggle expressions, the key parameters and their corresponding expressions to be edited through UI, etc. Then the default subtype and alterations are set as the



Std Component Library

	(a)	(b)	(c)
Supplier	Misumi	Misumi	Misumi
Category	sprue bushes	sprue bushes	sprue bushes
Major Type	sbtm	sbtm	sbtm
Sub-type	sbtm	sbtm	sbtm

Alterations

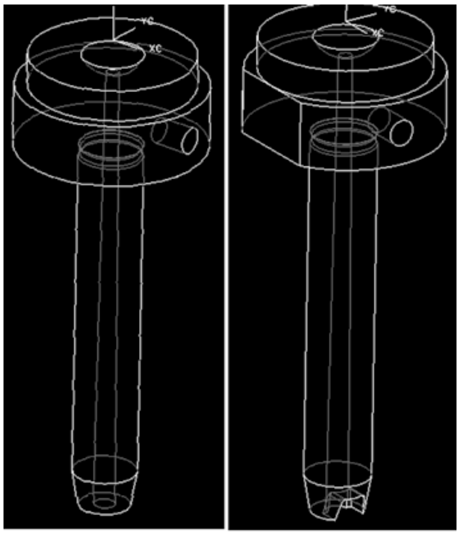
(a) ☐ KC ☐ AIW ☐ ATW ☐ AXW ☐ ALW ☐ BIR ☐ BTR ☐ BXR ☐ BLR ☐ CIQ ☐ CTQ ☐ CXQ ☐ CLQ ☐ ZC ☐ LKC

(b) ☑ KC ☐ AIW ☐ ATW ☐ AXW ☑ ALW ☐ BIR ☐ BTR ☐ BXR ☐ BLR ☐ CIQ ☐ CTQ ☐ CXQ ☐ CLQ ☐ ZC ☑ LKC

(c) ☑ KC ☐ AIW ☐ ATW ☐ AXW ☐ ALW ☐ BIR ☐ BTR ☐ BXR ☐ BLR ☐ CIQ ☐ CTQ ☐ CXQ ☐ CLQ ☑ ZC ☐ LKC

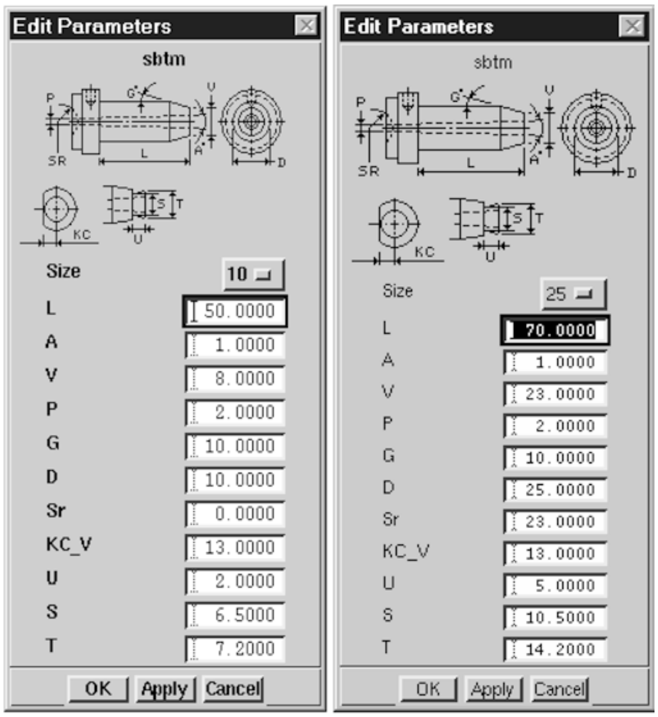
(a) (b) (c)

Fig. 3 QuickMould SCL main UIs



(a) (b)

Fig. 4 An example SCL part with different alterations



Edit Parameters — sbtm

	(a)	(b)
Size	10	25
L	50.0000	70.0000
A	1.0000	1.0000
V	8.0000	23.0000
P	2.0000	2.0000
G	10.0000	10.0000
D	10.0000	25.0000
Sr	0.0000	23.0000
KC_V	13.0000	13.0000
U	2.0000	5.0000
S	6.5000	10.5000
T	7.2000	14.2000

(a) (b)

Fig. 5 SCL parameter editing UIs

current choices. At this stage the user can toggle feature configurations. The exclusion rule among members of a configuration group applies. When an alteration is toggled on, other alterations in the same group are toggled off. Upon the user's selections, displayable and non-displayable parameters, constraints and tolerances, and additional attributes, such as available material, are assigned as object properties.

Second, the CAD template file is then loaded and saved as a new part file in the project directory (Fig. 4). The new feature configuration of subtype/alterations is effected immediately based on the user's selections. Sizing information, such as allowed sizes, parameter values and ranges, is obtained from the dimension data file and the key parameters in the "QM_STD_COMP" object are assigned with the predefined values for the default size. The user needs to select a size when adding a new component. If the user selects a different size, the predefined parameter values for the selected size are obtained from the dimension data file. They are assigned to the key parameters of the "QM_STD_Comp" object. When the user confirms the selection, the geometry dimensions are updated using QM_STD_Comp methods. The part is then inserted into the mould assembly as a component under a selected parent. From here, the user can edit, accept or delete this component as he wishes. When the user confirms this component and exits the SCL module, the object's attributes are stored as CAD attributes and the QuickMould object is deleted.

Up to now, the standard component can be seamlessly stored as a CAD component. As indicated by the UI shown in Fig. 5a, key parameters can be further edited or customized. When the UI appears, the "QM_STD_COMP" object has already established all the attributes related to this particular component. The size, and other key parameters in the "QM_STD_COMP" object are assigned with the current effective values. At the same time, constraints, parameter ranges are also retrieved from the dimensional data file. When the user keys in his input, the callback functions of the individual editing fields verify the input values against the constraints and ranges. If there is no violation, these input values are accepted as the object member attributes. Once the user clicks "Apply" or "OK" buttons from the UI, those expressions related to the key parameters of the "QM_STD_COMP" object are assigned with the new values, and the geometry is updated automatically.

8.2 Editing a standard component

A user can edit an inserted component within QuickMould application. Once it is activated, it searches all QuickMould objects (UDOs) in the assembly CAD files. They are registered in the currency dictionary. It means that Quickmould recognizes a selected standard component automatically, and instances a new "QM_STD_Comp" object. All attributes associated are retrieved to update the object. Then the main SCL UI as shown in Fig. 3a or 3b is established after searching the available library items. Actually, when the "QM_STD_COMP" is activated, its feature configuration file is read again, the information for the current configuration and the respective selection options for buttons are initiated in the UI as well. This standard component can now be edited exactly as it was the first time when it was inserted. Since the object has been fully established, the user can re-select "supplier", "category", "subtypes" and alterations, and edit size and key parameters, and then click "Apply". It is worth mentioning that when the category is changed to a different one, a new template is then loaded and a new "QM_STD_COMP" object is established. The original component template, configuration, and data files are not deleted until the user confirms the replacement.

8.3 Deleting a standard component

Methods have been implemented to delete selected standard components from the project assembly, so are the template, configuration and data files from the current project directory. Simultaneously, the related currency object and dictionary pointer in the session are also cleaned.

9 A case study

In this section, the component category mentioned above, "sprue bushings" as shown in Fig. 1, is studied to illustrate the functions of this component library. To add this category into the library, a configuration file is first created. It specifies the five major types, from "sbtm", to "sbtsh"; the corresponding CAD template file, "sbt_.prt"; dimensional data file, "sprue_bushes.dat"; and all bitmap files (*.bmp) required by UIs. Because there are no subtypes for "sprue bushings", by default, the effective major type is listed as the subtype, i.e. "sbtm"..."sbtsh". All the attributes are organized according to their scope in the configuration file, for example, "D_tolerance = h6" is applicable to all cases, while "material = HPM1" and "hardness_value = 40 HRC" are defined in the scope of major types "sbtm" and "sbtmh" only. This category has 15 alterations. They are divided into three groups in the configuration file, i.e. "KC" alone as one, "AIW" to "ZC" as another, and "LKC" as the third. Then, in the same file, parameters are grouped as displayable, like "L" and "SR", or non-displayable, like "OD" and "α". Their corresponding expression names are specified side by side. Constraints are also included in the format as "$V >= \alpha + 2$". They are evaluated every time when the CAD model is updated. Similarly, tolerances are represented as "H1 0.05 0.03", which can be used for process planning and tolerance analysis if it is integrated with a system as introduced in [12].

Next, a parametric CAD model is constructed with each feature configuration controlled by an expression. In this case, from "AIW" to "LKC" are optional alterations and each of them is controlled by an expression to suspend or to activate. Then, the dimensional data file is created and organized according to sizes in a dimensional table. Each size contains the predefined parameter values as specified in Fig. 1. Next, for each subtype or alteration, a sketch bitmap file with key dimensions and another without them are created and then incorporated into the library UIs automatically. Up to now, a library item, "sbt_" has been created.

Upon entering SCL module environment, the library recognizes this item automatically by searching the library directories. Fig. 3a shows the main SCL UI with this component item selected. The user has selected "Misumi" from the first pull-down menu out of other vendors, and then "sprue bushes" as the category. Dynamically, available major types, subtypes, available alterations and a default configuration are displayed in the UI frame (Fig. 3a). The effective configuration bitmaps are displayed on the UI as well. Once the user clicks "OK" or "Apply", the library element is loaded. The initial component model is shown in Fig. 4a. All the major/sub types are exclusive, for example, if the feature configuration for "sbtm" is activated, then others are suspended immediately. This exclusion is also applied to the alterations in the same group. For example, the user can select alterations "KC", "ALW" and "LKC" at the same time in the configuration (see Fig. 3b for the UI and Fig. 4b for the CAD model), but among alterations from "AIW" to "ZC", only one of them can be in effect at a time. As shown in Fig. 3c, on the UI, when "ZC" is selected, "ALW" is turned off automatically because they belong to the same group. "KC" and "ZC" alterations can be selected together because they are from different alteration groups (see Fig. 3c for the UI and Fig. 6a for the CAD model). When the user clicks the "Edit Component Parameters" button, the editing UI is displayed as shown in Fig. 5a; parameter values are extracted and displayed according to the CAD model. It can be seen that only displayable parameters like "L" and "SR" appears. Parameters like "OD" and "α" in Fig. 1 are not editable. At this stage, if the user accepts

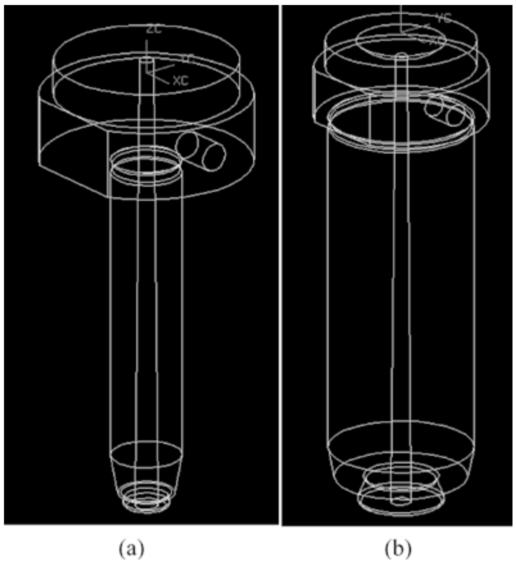


Fig. 6 An example SCL part with different parameter values

all the parameter values, by clicking "Apply" or "OK" button at the bottom of the UI, this component is added to the project assembly automatically. Then the UI as shown in Fig. 3c is returned. The user can still edit all the parameters by clicking the "Edit Component Parameter" button, which leads to the UI shown in Fig. 5a again. When the user changes the size from "10" to "25", then the library fetches the corresponding parameter values from the dimensional data file and updates the menu as shown in Fig. 5b. Again, upon the user's confirmation, the CAD model is updated as displayed in Fig. 6b. If the user has finally accepted this component, the library can generate order code like "SBTM 25-70.0-SR23-P2.0-A1.0-V23.0-G10-KC13-U5-S10.5-T14.2-end" automatically and store it as an attribute in the CAD model for future references.

10 Benefits and limitations

This component library is capable of handling topological configurations and compressed CAD files required for a catalogue. In comparison, traditional 3D parametric libraries develop a parametric model for each geometry topology and the parametric variations are restricted to size dimensions only [1, 2, 11]. Other more advanced 3D component libraries [12, 18, 19] do not allow topological configurations to be easily modified. Our component library is a significant improvement because it reduces the large number of CAD model templates. Take Misumi mould components catalogue [24] as an example. It contains a total of 195 major types and subtypes, and for each of them, there could be many alterations. If we assume 10 alterations for each major type or subtype on average, then the total number of CAD models required is 1950. In QuickMould, each component category is implemented as one item in the library; only 21 items are required for the same catalogue. It has been successfully used to define more than 200 different categories of industrial components from 5 vendors for commercial applications. This library can be easily reused and expanded with user-defined components without programming.

One limitation of the current implementation is that the library is currently only workable with one CAD platform only. A research project to develop a neutral CAD component library is currently underway. Another limitation is that an interface for advanced reasoning has not been developed. The flexibility in expanding object definitions allows different systematic enriched attributes to be easily incorporated, such as embodiment or multibody system analysis for conceptual mechanisms [17, 18, 19], or coding and clustering [12, 16]. This will be a direction for future research. Finally, the library model can be extended to provide a distributed library over the Internet. To ensure the scalability, an XML based object-oriented database needs to be built to support the library.

11 Conclusions

This paper has described an efficient object-oriented library model for defining mechanical components parametrically. The model extends current parametric methods by incorporating different geometric topologies and non-geometric information. This is an improvement on conventional parametric design where CAD models are hard-coded to deal with such information. This concise and application-oriented model is generic and capable of defining many kinds of mechanical components.

Acknowledgements QuickMould research project was sponsored by the InfoComm Development Authority (iDA) of Singapore and jointly developed by Nanyang Technological University (NTU) and Singapore Institute of Manufacturing Technology (SIMT).

References

1. Ezz AA, Al-Medfa O, Hamed M (1992) Design of an AutoCAD tablet menu for machine design. In: Computers in Engineering, Proceedings of the International Computers in Engineering and Exhibition, 2–6 Aug 1992, San Francisco, pp 281–286
2. Johannesson HL (1991) Computer aided part design based on standard component interface geometry. Adv Des Autom 32(2):347–352
3. Culley SJ, Webber SJ (1992) Implementation requirements for electronic standard component catalogues. In: Proceedings of the Institution of Mechanical Engineers, Part B: J Eng Manuf 206(3):253–260
4. Lodenstein MA, Romps DM, Tran P (1994) Development of mold design software. In: Proceedings, the Society of Plastic Engineers Annual Technical Conference (ANTEC 1994), Brookfield, CT, pp 1090–1093
5. Lye SW, Yeong HY (1992) Computer-assisted mould design for styrofoam products. Comput Ind 18(2):117–126

6. Tor SB, Lee SG, Chung YHSH (2000) A two-stage collapsible core for injection moulded plastic parts with internal undercuts. Int J Mach Tools Manuf 40(8):1215–1233
7. Fu MW, Fuh JYH, Nee AYC (2001) Core and cavity generation method in injection mould design. Int J Prod Res 39(1):121–138
8. Mok CK, Chin KS, Ho JKL (2001) An interactive knowledge-based CAD system for mould design in injection moulding processes. Int J Adv Manuf Technol 17(1):27–38
9. EDS Inc. (2000) UG Documentation Help for v18, Maryland Heights, MO
10. ThomasRegister.com,http://www.cadregister.com/home-text.asp
11. Cadalog,http://www.cadalog.com/index.php
12. Xue D, Dong Z (1997) Coding and clustering of design and manufacturing features for concurrent design. Comput Ind 34(1):139–153
13. Qamhiyah AZ (1998) A strategy for the construction of customized design libraries for CAD. CAD 30(11):897–904
14. Deng Y-M, Britton GA, Tor SB (1998) A design perspective of mechanical function and its object-oriented representation scheme. Eng Comput 14(4):309–320
15. Zhang WY, Tor SB, Britton GA (2001) A prototype knowledge-based system for conceptual synthesis of design process. Int J Adv Manuf Technol 17(8):549–557
16. Regli WC, Cicirello VA (2000) Managing digital libraries for computer-aided design. CAD 32(2):119–132
17. Keirouz W, Pabon J, Young R (1990) Integration parametric geometry, features, and variational modeling for conceptual design. In: ASME, Design Engineering Division, vol 27, Design Theory and Methodology, DTM'90 ASME Design Technical Conference, 16–19 Sep 1990, Chicago, pp 1–9
18. Thornton AC, Johnson AL (1996) CADET: A software support tool for constraint processes in embodiment design. Res Eng Des 8(1):1–13
19. Daberkow A, Kreuzer EJ (1999) An integrated approach for computer aided design in multibody system dynamics. Eng Comput 15(2):155–170
20. Ong SK, Prombanpong S, Lee KS (1995) An object-oriented approach to computer-aided design of a plastic injection mould. J Intell Manuf 6(1):1–10
21. Deng Y-M, Britton GA, Lam YC, Ma Y-S (2001) A feature-based CAD-CAE integration model for injection molded product design. In: Proceedings, the 16th International Conference on Production Research, Prague 29 Jul–3 Aug 2001
22. Gan PY, Lee KS, Zhang YF (2001) A branch and bound algorithm based process-planning system for plastic injection mould base. Int J Adv Manuf Technol 18(9):624–632
23. Yarlagadda PKDV, Ang CTK (2001) Development of a hybrid neural network system for prediction of process parameters in injection moulding. J Mater Process Technol 118(1–3):110–116
24. MISUMI Corporation (1993) Face standard components for plastic molds. May 1993–Apr 1996, 2-Chome, Toyo, Koto-Ku, Tokyo
25. Britton GA, Ma Y-S, Tor SB (1999) Object technology development and unigraphics. In: Proceedings, Unigraphics User Group 1999 Spring Conference: Manage design evolution, Newport Beach, CA
26. Britton G, Tor SB, Wang Y (2000) Virtual concurrent product development of plastic injection mould. In: Proceedings of the Institution Mechanical Engineers, Part B: J Eng Manuf 214(2):165–168