Defence Science Journal, Vol. 57, No. 1, January 2007, pp. 133-138
© 2007, DESIDOC

SHORT COMMUNICATION

Database-centric Development of Menus and Graphic User Interfaces

R.B. Aggarwal, Amit Dhawan, and Jay Shankar Kumar
Institute for Systems Studies and Analyses, Delhi-110 054

ABSTRACT

The database-centric approach to graphic user interface (GUI) development, quickly and easily manages standardisation and modification of labels and look and feel of controls by keeping various control-creation data into the database. The runtime generation of controls provides the flexibility to control their creation and modification issues. This method freezes the application code once the development is over. The process of recompilation is eliminated when creation or modification of controls is done. Dynamic controls such as menus, label, text box, button, combo box, list box, group box, check box, radio button, tab control, spin button, tree control can be easily formed and controlled using this approach.

Keywords: Graphic user interface, dynamic controls, database, menu, hierarchy code, link table

1. INTRODUCTION

The traditional way of developing graphic user interfaces[1] is a straightforward method of WYSWYG (What You See is What You Get). A tool box is used for drag and drop of controls on the screen. The menu items are arranged on the menu bar by clicking on the menu editor. The moment any modification is done on the screen, the whole application needs to be recompiled.

A deeper insight into this approach shows that it poses great difficulties when a request for change is made. Such requests may be, for example, change in menu items, changes in their sequencing on screen or change in their types. Since the entire set of menu items are hard-coded, it is very difficult to make all the changes on the fly. This leads to opening the source code and making changes not only to the menu item, but also in the associated identifications and links to the associated callback functions, leading to difficulties in software maintenance.

When one talks about screens, the problems associated are even more complex. There may be request to change a label which has been used in several screens. This leads to the change of labels at all the places where it occurs. This is a cumbersome task since one has to navigate through various front end screens for implementing the changes, leading to poor quality of software. Many a times, there is a request to change the field type from integer to string. Here, the screen has to be virtually redesigned. The difficulty increases manifold if the software deployment is carried out in geographically different locations and the change request has to be executed at several places.

There may be a request to make an application having *n* number of text box controls. The code

Revised 22 March 2006

has to be duplicated *n* times, for the same type of control. The traditional approach increases the code size as the number of controls increases, leading to duplication of code in the application.

To overcome these difficulties, an approach for development of menus and graphic user interfaces using a database-centric solution is proposed. It will lead to faster development of applications.

2. DATABASE-DRIVEN APPROACH FOR MENUS/GRAPHIC USER INTERFACES

The approach has been implemented for the following three related aspects:

Label: To centrally control the labels in menus and screens, each label is given a unique code. The code is used as a part of procedure, whereas label description is used for display on the menu or screen. The reusability of label increases as these are standardised using this approach. Moreover, user-specific installation can be configured.

Menu: All the menu items are placed in a database table. One row of records is required to create a new menu. Each menu is assigned a hierarchy code. This specially designed code links the position of menu in different sequences of occurrence. This approach facilitates centralised control of a large number of menus and submenus at different levels. A convention for hierarchy code is followed. The first menu at each level starts from 01 followed by others, i.e., 02, 03, and so on, depending on the number of menus appearing initially. The submenus to the main menus are coded as 0101, 0102, 0103, and so on. As an example, the third level of submenu can be coded as 06020304 which correspond to the highlighted menu of Fig. 1.

06 - Top-level menu position

02 - First-level submenu position

03 - Second-level submenu position

04 - Third-level submenu position

Re-sequencing of menus can be done by changing the hierarchy code. To modify a menu item, menu name in the table is changed and application is run. The changed menu is reflected in the application. The name of the procedure to be called on the leaf menu item is also stored in the table, leading to higher order of reusability of the procedure written in the application.

Screen: A graphic user interface may contain various types of controls such as label, button, text box, combo box, etc. These controls–creation mechanism is stored as a procedure in the library. Each control item has several attributes, e.g., position on screen, size, field type, display characters, etc. All the control-related information is placed in a table. One row of records is required for each control feature. Position of control is adjusted from the X and Y coordinates of the top left corner of the screen. The size is controlled from control length and control width values. These parameters are being shown in Fig. 2 for a text box control. The default value and scroll size (number of characters) can be set and modified for text box control. The

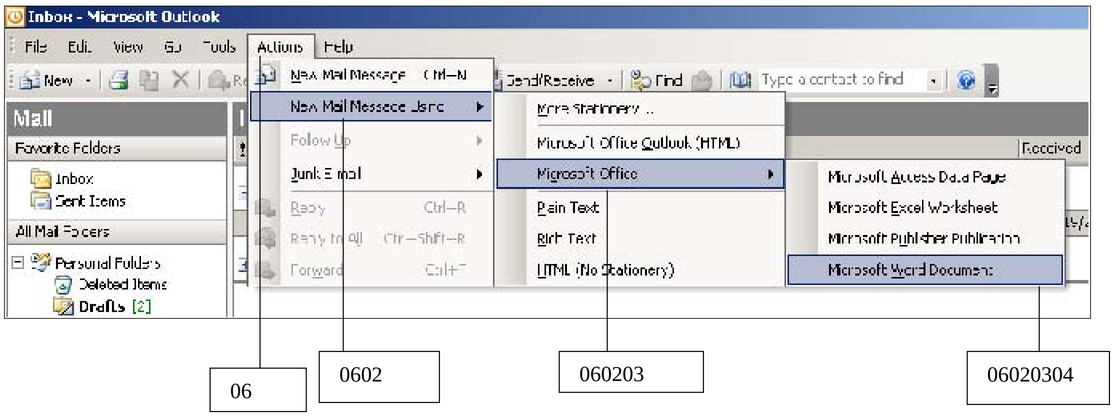


Figure 1. Hierarchy code structure for dynamic menu design.

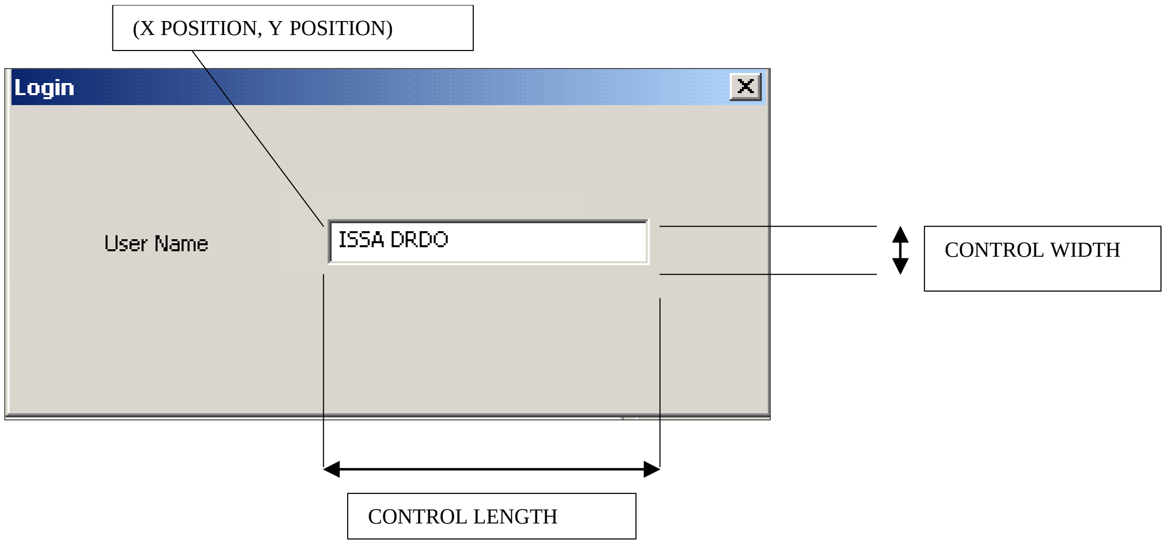


Figure 2. Position and dimension of control for dynamic design.

controls like combo box and list box can be populated with user chosen table data by mentioning the name of the table. The field type of text box can be set either as string or as an integer.

3. FLEXIBLE PART TO BE STORED IN THE DATABASE

As stated earlier, database plays an important role in this approach. The data structure along with some sample data for implementation is given in Tables 1, 2, and 3. The menu name and hierarchy code data in the menu table corresponds to the highlighted menus, as shown in Fig. 1.

Apart from the tables storing flexible part, various libraries have been used as stated below:

(a) Control library–where all the control-creation codes have been written

(b) Application library–used to call the control library function

(c) Menus as executable file (.exe)–a main program where menu creation code has been written.

Table 1. Menu table description

MENU_NAME	MENU_CODE	HIERARCHY_CODE
Actions	139	060000000000
New mail message	140	060200000000
MS Office	152	060203000000
MS Word document	168	060203040000

Table 2. Label table description

LABEL_NAME	LABEL_CODE
Junk e-mail	62
Plain text	16

Table 3. Screen table description

SCREEN_CODE	CONTROL_TYPE	CONTROL_ID	LABEL_CODE
957	Label	1271	249
957	Textbox	1273	0
X POSITION	**Y POSITION**	**CONTROL_LENGTH**	**CONTROL_WIDTH**
10	10	200	20
150	10	100	20
SCROLL_SIZE	**FIELD_TYPE**	**DEFAULT_VALUE**	**LINK_TABLE**
0	Null	Null	Null
8	Number	Select	Null

The menu item calls the application library, which in turn calls the control library.

4. BENEFITS ACCRUED USING THIS APPROACH

The approach has been designed in a way to provide some of the following benefits:

- *Labels codified to centrally update the label without changing each GUI and recompilation of source code*: All the labels are stored in a separate table with corresponding label code which is used to form specific controls like buttons, check boxes, radio buttons, etc. The centralised control of labels helps in easy modification. A single change in a label is reflected at all the places where it occurs.

- *Multi-layer library created to increase the reusability for the code*: The code has been written in the form of a library which can be used anywhere and any number of times. The library comprises different segments for different types of control. Any number of controls can be generated with the fixed volume of the code.

- *Software management-related tables loaded in the RAM at initiation of application to avoid multiple data access*: The data relevant to the particular screen being displayed are fetched from the table and loaded in RAM. So, there is a one-time database I/O which is used for further manipulations.

5. CONSTRAINTS OBSERVED

The approach is quite robust. However, some constraints observed in the process are as follows:

- *Close knitting with database*: Since placing of control requires pixel values on the screen, the exact position can be found only by some hit and trial of pixel values in the database.

- *Longer time spent in initiation of application*: If a large number of data is needed for a particular screen, the database I/O consumes more time and the application may become slower.

6. CASE STUDIES

6. Land Weapon Performance Evaluation Menu

The data-centric, dynamic menus have been generated which can pop up to four submenu levels. A sample is given in Fig. 3. The analysis domain is grouped to perform four distinct functions, namely, 'Weapon Evaluation', 'Force Multiplier', 'Force Potential', and 'Weapon Deployment'. Weapon Evaluation can be performed by four proposed approaches like expert opinion, simulation, empirical, and analytical. Further, simulation can be carried out in four steps of 'Session Option', 'Weapon Selection', 'Scenario Template', and 'Simulation Control Parameters'. To create a 'Scenario Template', four issues of 'Environment Setting', 'Orbat Generation', 'Force Deployment', and 'Combat Plan Preparation' have been considered.

6.2 Adaptive Dynamic Model Screen

Adaptive dynamic model (ADM) is an analytical approach to perform weapon evaluation. Dynamic controls are used for every screen in ADM module. A sample screen shot is given in Fig. 4. This screen is used to organise the red and blue forces by selecting arm type, weapon category, weapon name,

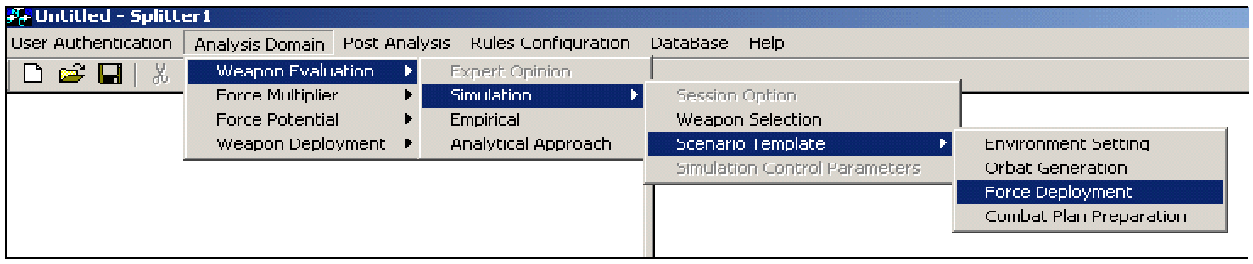
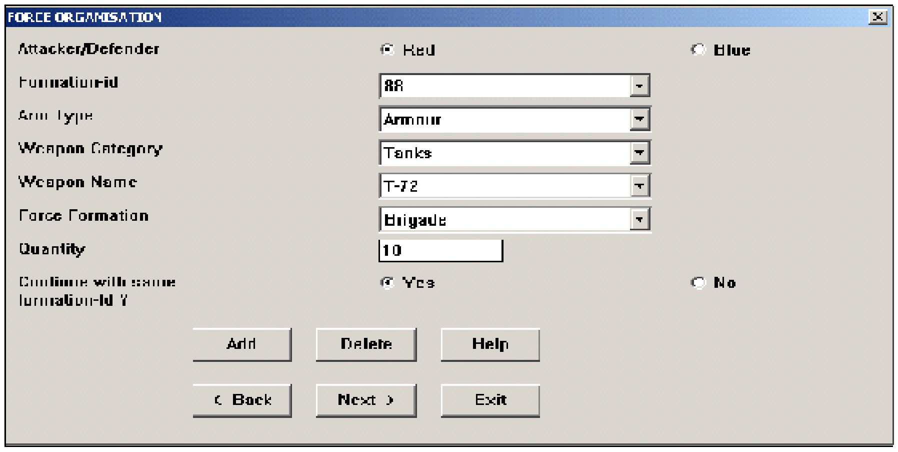


Figure 3. Dynamic menu implementation



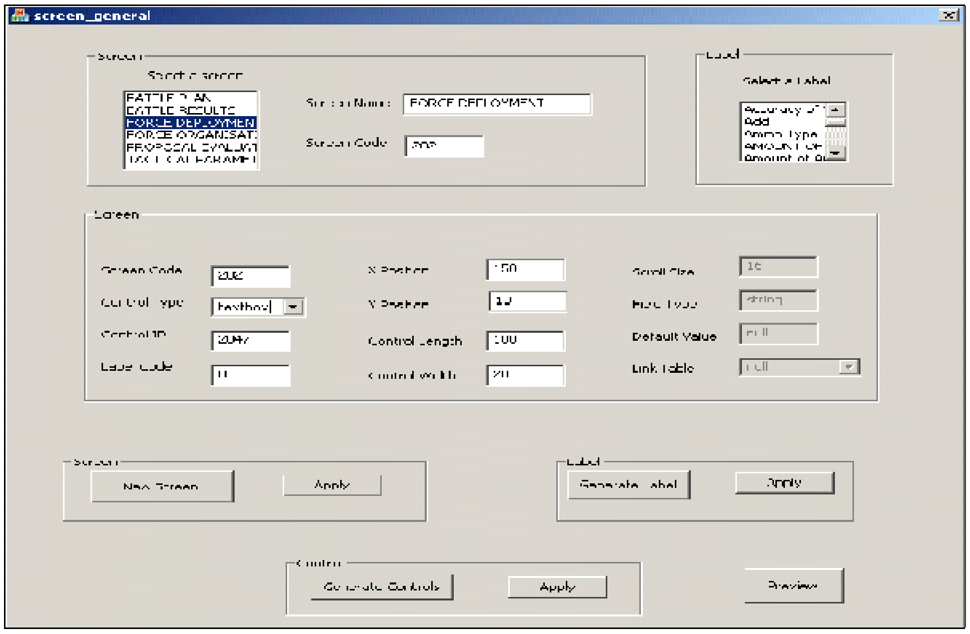
Attacker/Defender	[✓ Red]	Blue
Formation-id	88	
Arm Type	Armour	
Weapon Category	Tanks	
Weapon Name	T-72	
Force Formation	Brigade	
Quantity	10	
Continue with same formation-Id ?	[✓ Yes]	No

Figure 4. Dynamic controls implementation.

and force formation and entering number of fighting formations. An implementation of labels, radio buttons, text boxes, combo boxes and some additional buttons is displayed in this screen. This screen has been developed using the approach outlined for labels and screens.

6.3 Static Data Management

This requires large data entry for creating multiple screens. To automate this data entry, a GUI is used which handles all the data required to form the specific control and stores these in a screen table just by click of the mouse. The screen shot for this purpose is given in Fig. 5. This screen uses three different groups, namely, 'New Screen', 'Generate Label', and 'Generate Control' along with their 'Apply' button. One can place the control either on a 'New Screen' or can add to an existing screen. To create a label, a particular label is selected from the list of labels and 'Generate Label' button is used. The default values are set in the screen parameters. Then 'Apply' button is used to store all the parameters in the database. The 'Generate Control' button is used for creating controls other than labels. The 'Preview' button is used to see the created controls on the fly.



BATTLE PLAN
BATTLE RESULTS
FORCE DEPLOYMEN
FORCE ORGANISATI
PROPOSAL EVALUATI
TACTICAL PARAMET

Screen Name: FORCE DEPLOYMENT
Screen Code: 202

Accuracy of
Add
Ammo Type
AMOUNT OF
Amount of A

Screen Code	202	X Position	150	Scroll Size	16
Control Type	textbox	Y Position	17	Field Type	string
Control ID	2047	Control Length	100	Default Value	null
Label Code	11	Control Width	20	Link Table	null

Figure 5. GUI to create dynamic screen.

7. COMPARISON WITH TRADITIONAL APPROACHES

By comparing various aspects of the traditional approach, vis-à-vis the proposed approach, one can assess the strength of this approach as presented in Table 4.

Table 4. Comparision between traditional and database approach of GUI development

Traditional approach	Database approach
Requires tool box for drag and drop of controls	Does not need tool box
Opens editor for any modification	Does not open editor for modification
Requires recompilation	No need of recompilatior
Changes to the label names are local	Changes to the label nam are global

8. CONCLUSION

The above strength of the database-centric approach leads to various advantages. It is found helpful in:

(a) Time saving in modification and sequencing of label names

(b) Faster development

(c) Better quality control

(d) More manageable

(e) Eliminating hard coding.

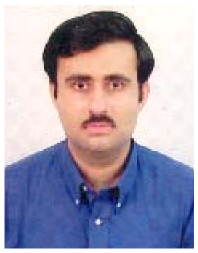
REFERENCES

1. Kruglinski, David J. Programming Visual C++'. 109 p.

2. Schildt, Herbert. MFC programming, Ed. 2. Tata McGraw Hill, 2000. 467 p.

Contributors

Mr R.B. Aggarwal obtained his MSc (Mathematics) from the University of Delhi in 1974. He joined DRDO in 1976 and presently working as Scientist F at the Institute for Systems Studies and Analyses (ISSA), Delhi. He has developed various management information systems (MIS) packages in the area of personnel information system, project management, activity plan and monitoring of progress, etc. He is presently working on project Land Weapons Performance Evaluation. He has developed systems analysis/scientific software in the area of optimal force mix, ionospheric area prediction, etc.



Mr Amit Dhawan obtained his BTech (Computer Science) from the Aligarh Muslim University in 1997. He joined DRDO in 1998 and presently working as Scientist D at the ISSA. He is qualified internal quality auditor from IIQM, Jaipur. His area of interest is: Global information system and databases.

Mr Jay Shankar Kumar obtained his BTech (Computer Science) from the National Institute of Technology (NIT), Jamshedpur, in 2002. He joined DRDO in 2002 and presently working as Scientist C at ISSA. He has developed various mathematical models using Visual C++.NET and Oracle. His areas of research are: Software design and development, database management and crystal reporting.