Available online at www.sciencedirect.com
SciVerse ScienceDirect
Procedia Engineering 29 (2012) 584 – 588
Procedia Engineering
www.elsevier.com/locate/procedia

2012 International Workshop on Information and Electronics Engineering (IWIEE 2012)

Design and Implementation of Reusable Components Using PowerBuilder

Zhenjun Tang*, Xianquan Zhang, Liyan Huang

Department of Computer Science, Guangxi Normal University, No.15 Yucai Road, Guilin 541004, P.R. China

Abstract

Component technology is a key technology of software reuse. This paper investigates PowerBuilder based technology of software reuse, especially the technology of component design. To build a reusable component, reusable elements in the application system are firstly extracted. The reusable components are then used to form a reusable component library. When designing application system, suitable components are selected from the reusable library and then instantiated. Software system is implemented by composing the instanced components under a reusable framework. Practical results show that the use of reusable components can improve the efficiency of software development.

Keywords: Software reuse; Component; Object-oriented technology; PowerBuilder (PB); Datawindow

1. Introduction

Software reuse is an efficient way to improve software quality and productivity [1]. It can avoid repeated works and therefore improves efficiency of software development and reduces the time and expenditure. Moreover, it makes the software reliable and easy to maintain [2]. In fact, software reuse has been considered as an important solution to solve software crisis and improve software quality and productivity. Generally, software reuse is to use the existing software or software knowledge to develop new software [3], where the existing software and software knowledge include documents, data and programs, e.g. results of requirement analysis, source codes and test plans. Component technology is an important part of software reuse and attracts many researchers' attentions in the past decades.

* Corresponding author. Tel.: +86-15295990968.
E-mail address: tangzj230@163.com, zjtang@gxnu.edu.cn.

doi:10.1016/j.proeng.2012.01.008

Components are those modules, which can be clearly defined in the application system. Reusable components are the components with independent function and reusable value [4]. This paper investigates PowerBuilder (PB) based technology of software reuse, especially the technology of component design.

2. PB-based technology of software reuse

Object-oriented technology is a popular technology in software engineering and is proved to be a successful technology of software reuse [5]. Its concepts and principles supporting software reuse include object, class, abstraction, encapsulation, inheritance, general-special structure, aggregation, granularity control, polymorphism, and so on. PB is a popular tool with visual integrated development environment for developing database application systems. It supports relational database management system and object-oriented development method. Class is the basis of object-oriented technology and also an important way of software reuse. In PB, class, also called object type, can be classified into two categories: (1) Standard objects, e.g. application, window, menu and function, (2) User-defined objects. Object is the encapsulation of data and operations. The object data includes attributes and user-defined instance variables, whose access properties can be one of the following values: public, protected, and private. The object operations consist of functions and events, where the functions can be divided into two types: object function and user-defined function. Software reuse in PB is often achieved by reusing object type [6]. Generally, reuse in PB can be divided into two kinds:

(1)Direct reuse: It is achieved by declaring an instance of an object type. For example, there is a window object called ‘w_user’ in the reusable library. To reuse this object, we can declare instances as follows: w_user *myuser*1, *myuser*2.

(2)Inheritance reuse: Inheritance is an important feature of object-oriented technology and also a powerful mechanism for software reuse. It can be done as follows. New object types are firstly defined as the derived objects of existing objects. This is to inherit characteristics of the existing objects. In other words, the existing objects are reused. To achieve a particular function, we need to define specific operations or instance variables in the new object types. PB supports the inheritance of windows, menus and user objects.

3. Reuse design

Process of reuse design is composed of four steps: abstraction, selection, instantiation, and integration [7]. Abstraction can be divided into abstract specification and abstract implementation. It is to extract reusable objects from the specific program language, machine and other environmental details. In general, each abstraction can describe a related reusable object set. Conversely, the related reusable object set can also determines the abstraction. Obviously, abstraction is a key technique of software reuse. It determines the possibility of reusing the extracted components. The higher the abstraction level, the bigger the possibility of reusing the components. In practice, one should first determine the reusable parts of an application system. This can be done by domain analysis method [8]. Take the management information system for example. As each system has to exchange data with other external system, functions of data export and data import are the common parts of these systems. Therefore, two reusable components, i.e. ‘Data Import’ and ‘Data Export’, can be abstracted. Similarly, we can discover other reusable components, e.g. ‘Print’ and ‘Search’.

When we use PB to design reusable components, abstraction is to find reusable objects, such as windows, menus, functions, data windows and so on. To make reusable components as independent as possible, we need to abstract source codes. In other words, specific objects, e.g. the name of a specific data window, cannot appear in the source codes. If business-related objects are needed, we can declare

local instance variables instead of specific objects. In addition, when using object's functions, we should select those functions unrelated to field name. For example, to get a string in a specific row and column of a datawindow, we should use the function GetItemString(long row, integer column) instead of GetItemString (long row, string column). Similar functions include GetItemDate (long row, integer column), GetItemTime (long row, integer column), GetItemNumber (long row, integer column) and so on.

Selection means searching and comparing reusable components in library to obtain reusable components satisfying application requirement. It refers to component classification, retrieval and description. For PB, reusable components are stored in the library with suffix name '.pbl'. In practice, we use the library function 'export' to output components to external files. When reusing the components, we exploit the function 'import' to import the external files into the object system.

Instantiation is the technique of passing, converting, or constraining parameters for reusable objects. By instantiation, reusable objects satisfying the problem specifications are converted to executable codes. In PB, instantiation usually refers to direct reuse of object.

Integration is the process of building a software system by composing instanced reusable components under the guide of a reusable software framework. In fact, integration is an optional step. If the system framework is not provided by reusable technology, integration is not needed.

4. Instance design and implementation

To show the process of reuse design in PB, we give an example here. This example is about a reusable component for print. The process is generalized and suitable for other components.

4.1. Abstraction

Print is an essential function of each application system. It is a typical reusable element. Generally, print functions include: printer setup, page selection, copy selection and so on. These functions are the common information of application systems. So we can abstract the common information to build a reusable component. To do so, we design a print window called 'w_print'. Layout of the window and names of the controls in the window are shown in Figure 1. Properties of the window are as follows: title name is 'Print', 'WindowType' is 'Response!', 'WindowState' is 'Normal!', those values of 'Visible', 'Enabled', 'TitleBar', and 'ControlMenu' are all 'true'. Clearly, the window 'w_print' is a reusable object. The print functions are done by manipulating the datawindow. To achieve reuse, we declare a local instance variable whose type is 'datawindow' instead of using the name of the datawindow control.

This component has been used in real application systems and works well. The codes are compiled and tested in PB 7.0. Detailed source codes of the print window are given as follows.

(1) Declare two local instance variables in the window

```
datawindow i_dwToActOn  // store the data window object passed from other modules
string i_szFileName       // name of print file
```

(2) Source codes in the event 'open' of the window

```
string szCopies
i_dwToActOn=Message.PowerObjectParm // Receive the object of data window passed from other modules.
st_current_printer.text=″ Current printer:″+string(i_dwToActOn.object.datawindow.printer) // Get default printer.
szCopies=string(i_dwToActOn.object.datawindow.print.copies)  // Get default number of print copies.
if szCopies<>"" and szCopies<>"0" then  em_copies.text=szCopies
else   em_copies.text="1"  end if
cbx_collate.checked=(Upper(string(i_dwToActOn.object.datawindow.print.collate))="YES")
i_szFileName=trim(string(i_dwToActOn.object.datawindow.print.FileName)) // Get default print file name of the data window.
cbx_print_to_file.checked=(i_szFileName<>"")
```

(3) Source codes in the event 'clicked' of the control 'rb_pages'

```
if this.checked then sle_page_range.SetFocus() end if
```

(4) Source codes in the event ‘clicked’ of the control ‘cbx_print_to_file’

```
String szFile
if this.checked then  GetFileSaveName("Please select the file name ",i_szFileName,szFile,"PRN","PrintFile (*.PRN),*.PRN")
else i_szFileName="" end if
```

(5) Source codes in the event ‘constructor’ of the control ‘ddlb_range_include’

```
this.selectItem(1)   // Select the first option in the drop-down list box
```

(6) Source codes in the event ‘clicked’ of the control ‘cb_printer_setup’

```
Printsetup()     // printer setup
st_current_printer.text= " current printer: "+string( i_dwToActOn.object.datawindow.printer)
```

(7) Source codes in the event ‘clicked’ of the control ‘cb_cancel’

```
close(parent)   // close window
```

(8) Source codes in the event ‘clicked’ of the control ‘cb_ok’

```
integer nIndex
String szModify,szPage,szReturn
if isnull(em_copies.text) or em_copies.text="" then   MessageBox("Tip"," Please input the copy number.")
   em_copies.setfocus()  end if
szModify="DataWindow.Print.Copies="+em_copies.text  // Modify the copy number
if cbx_collate.checked then  szModify=szModify+"DataWindow.Print.Collate=Yes"
else szModify=szModify+"DataWindow.Print.Collate=No" end if
if cbx_print_to_file.checked then     szModify=szModify + "DataWindow.Print.FileName=" + i_szFileName
else    szModify=szModify + "DataWindow.Print.FileName=""  end if
if rb_all_pages.checked then     // Set the print pages according to the selected page range.
        szModify=szModify + "Datawindow.Print.Page.Range=""
elseif rb_current_page.checked then
        szPage=i_dwToActOn.Describe("Evaluate('page()'," +String(i_dwToActOn.GetRow())+")")
        szModify=szModify + "DataWindow.Print.Page.Range="'+szPage+"'"
else   szModify=szModify + "DataWindow.Print.Page.Range="'+sle_page_range.text+"'"  end if
nIndex=ddlb_range_include.FindItem(ddlb_range_include.text,0) // Get the selected index of the drop-down list.
szModify=szModify + "DataWindow.Print.Page.RangeInclude="+String(nIndex - 1)
szReturn=i_dwToActOn.modify(szModify) // Property modification
if szReturn<>"" then PopulateError(-1,szReturn) end if  //Give a tip if property modification failed.
parent.visible=false //Hide the print window
i_dwToActOn.Print(TRUE)  //Print the contents of the data window
this.setfocus()   close(parent)
```

4.2. Selection

The aim of selection is to import the print component into the current application system. This is achieved as follows. Select the print component from the reusable component library and use ‘export’ function to export it to an external file. Next, the external file is included to the current system by using ‘import’ function.

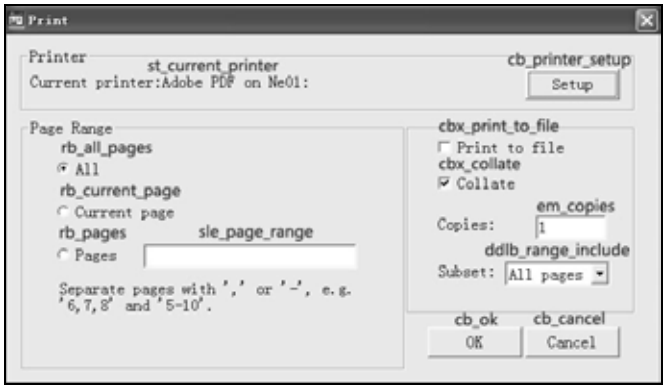

Fig. 1 Layout of the print window ‘w_print’

4.3. Instantiation

When using the print component ‘w_print’, we first declare an instance variable (myprint) of ‘w_print’ and then exploit the function ‘openwithparm’ to transmit the datawindow for print. Detailed codes are as follows.

```
w_print myprint    // Define an instance variable of ‘w_print’
openwithparm(myprint, dw_1)  //dw_1 is the data window for print.
```

4.4. Integration

Select reusable components and instantiate the components. If the software framework is provided by a reusable technology, the software system will be implemented by composing these instanced reusable components.

5. Conclusions

In this work, PB-based reusable software technology has been investigated. During reuse design, a key step is to abstract reusable components from the application system. The higher the abstraction level, the bigger the possibility of reusing the extracted components. Moreover, implementation codes of reusable components should be unrelated to specific objects. In general, specific objects are defined as local instance variables, whose values are obtained by passing parameters in the step of instantiation. Practices show that exploiting PB-based reusable components to develop software can shorten development time, improve development efficiency, and reduce cost.

Acknowledgements

This work was partially supported by the Natural Science Foundation of China (61165009, 60963008), the Guangxi Natural Science Foundation (2011GXNSFD018026, 0832104), the Project of the Education Administration of Guangxi (200911MS55), the Scientific Research and Technological Development Program of Guangxi (10123005–8), and the Scientific Research Foundation of Guangxi Normal University for Doctors.

References

[1] W. B. Frakes and K. Kang, Software reuse research: status and future, *IEEE Transactions on Software Engineering*, 2005; **31**:529−536.

[2] S. Wang, Z. He and K. Wang, Studies on software reuse technology, *Computer Engineering and Design*, 2000; **21**:10−15. (in Chinese)

[3] F. Yang, B. Zhu and H. Mei, Software reuse, *Journal of Software*, 1995; **6**:525−533. (in Chinese)

[4] Z. Lin and D. Yang, Software component reused technology overview, *Computer Engineering and Design*, 2004; **25**:877−880. (in Chinese)

[5] Y. Li and R. Zhang, The realization of software reuse, *Computer Engineering*, 1995; **21**:37−40. (in Chinese)

[6] X. Zhang, L. Jiang and Z. Tang, Software reuse research based on PowerBuilder, *Micro-computer information*, 2006; **22**:298−300. (in Chinese)

[7] Y. Chen, F. Li and Y. Wu, Studies on software reuse technology, *Transactions of Beijing Institute of Technology*, 1998; **18**:712−717. (in Chinese)

[8] H. Yu, Y. Song and W. Liu, Recognition of software reuse, *Microcomputer Development*, 2004; **14**:114−116. (in Chinese)