SPECIALIZED OBJECT-ORIENTED TOOLS FOR THE DEVELOPMENT OF INFORMATION-CALCULATING APPLICATIONS

Arslan Enikeev, PhD
Mahfoodh Bilal Ahmed Mohammed, MSc
Kazan Federal University, Russian Federation

Abstract

The paper presents an approach to building specialized object-oriented software tools for the development of so-called information-calculating applications including computer aided accounting, business correspondence, statistics etc . These software tools form an integrated development environment allowing the computer assisted development of information-calculating applications. This development environment consists of a formula interpreter, a screen form generator and a specialized library of classes. The implementation of all these components was carried out using the Visual FoxPro database system and has been practically tested on a series of commercial applications concerning computer aided accountancy and business correspondence.

Keywords: Software tools, programming technique, information-calculating applications

1.Introduction

One of the main features of the development of software products in the last decade is the emergence of a variety of efficiency improving applications of object- oriented technologies. Among them are CASE tools (Jacobson, 1994), design patterns (Gamma, 1995), investigations of object-oriented modeling languages (Eliëns,2000, DeLoach,2000) and many similar projects. Many of these projects have a generalized approach and they usually ignore or do not adequately represent the specifics of individual subject areas, forcing developers of specialized applications to use cumbersome and inadequate tools. On the other hand, excessively narrow specialization of technological tools in most cases condemns them to a single use, after which they are discarded. Therefore, we need to look for approaches that provide a flexible combination of versatility with specialization on the basis of class selection, which brings together many of

these specialized tasks, and allows the building of an integrated development environment for a set of related applications. This paper represents one such approach, focusing on the construction of software tools for the design and development of so-called information-calculating applications used for accountancy, financial analysis, office work, bank operations, statistics, etc. Distinguishing features of this class of problems are the processing of a set of tables and making calculations which can be conveniently represented by the tools of relational database systems.

2. The Specialized Integrated Development Environment

The concept of the integrated development environment is defined as a set of tools and techniques allowing the design and development of software applications. This concept, earlier used in procedural programming, has more recently proved to be a very important attribute for object-oriented programming. Integrated development environments traditionally include tools for the design and implementation of software systems, different libraries of classes, programming tools and program generators. We propose a new specialized integrated development environment for information-calculating applications. This environment consists of a formula interpreter, a screen form generator and a specialized library of classes. The implementation of all these components was carried out using the Visual FoxPro database system (Bazian, 2000) and has been practically tested on a series of commercial applications for accounting and business correspondence tasks.

2.1. The Formula Interpreter

Many of the information-calculating tasks are characterized by frequent changes in the methods of calculation (for example, a change in tax rates or wage indexation because of inflation, a change of accounting methods, etc.) Therefore we need parameterization of the calculating formulas by their separation from the program part in order to keep this part invariable with respect to the above changes. In this way, alterations to calculation methods require only changes of the calculation formulas, represented in the parametric environment for the formula- interpreter. The formula interpreter is defined as the function $FORM(S,F):S->S$, where $S=(T_1,T_2,\ldots,T_n)$ is a list of the related tables, T_i is the table from the list $S, i=1..n$, $F=(f_1,f_2,\ \ldots,f_k)$ is a list of formulas, where each formula $f_i(i=1..k)$ is represented by a tuple $<n,b,v,e>$, where n is a l number of the formula, b is a logical expression that defines the conditions for the applicability of the formula, v is the name of a variable or attribute (the left part of the formula), which is assigned the value of the expression e (on the

right-hand side of the formula.) The result of the formula's application can be interpreted as the statement " if b then v := e ". The implementation of the interpreter in the database system Visual FoxPro is based on the representation of a list of formulas, F in the form of a table with the structure defined by a tuple (the list of fields):

<P_ORDER(N, 5), P_COND(C, 60), P_OBJ (C, 10), P_FORM (C, 60), P_COMM (C, 80) >,

where P_ORDER is a number of the formula , P_COND is a Boolean expression defining a requirement for the applicability of the formula, P_OBJ is the name of the assigned object (the left side of the formula, defined as the field name or the name of a local variable), P_FORM is an expression that defines the right side of the formula and P_COMM is a comment. For implementation of the formula- interpreter we can apply the following procedure (it is presented in the Visual FoxPro language).

```
select T
go top
DO WHILE !eof()
    select F
    DO WHILE !eof()
        vcond=alltrim(P_COND)
        vobj=alltrim(P_OBJ)
        vform=alltrim(P_FORM)
        select T
        IF &vcond && checking the applicability conditions
          IF UPPER(substr(vobj,1,1))='X'
             &vobj=&vform
            * if assigned object is a variable
          ELSE
             replace &vobj with &vform
            *if assigned object is a field name
          ENDIF
        ENDIF
        select F
        skip
    ENDDO
    select T
    skip
ENDDO
```

This procedure is restricted by the following conditions: - for simplicity, we use here only one table as the object of interpretation, assuming that the increase in the number of tables will not affect significantly the nature of the algorithm; - if the name of the assigned object starts with the letter x then the object is recognized as a variable, otherwise as a field name; - formulas are calculated in numerical order.

2.2. Screen Form Generator.

The screen form generator works on the basis of the screen form specification. This is represented as a set of table structure specifications

associated with the corresponding screen form. The screen specification form is defined as $SPEC = \{S_{T1}, S_{T2}, \ldots, S_{Tn}\}$, where the set $\{T1, T2, \ldots, Tn\}$ consists of all the tables associated with the corresponding screen, and S_T is a specification of the structure of table T. The table structure specification is represented as a set of field specifications, $\{s_{f1}, s_{f2}, \ldots, s_{fk}\}$, where the specification of the field f is determined by the tuple $s_f =< n, t, l_1, l_2, r, d >$, where n is the name of the field, t is a type, l_1, l_2 define lengths, r is a caption and d is a definition domain for the field, f. The generator provides an automatic change of table structures and screen form structures on the basis of the relevant specification. In the Visual FoxPro database system, table structure specification is defined by a table with the following structure:

Field_name (C, 10) is the name of the field;
Field_type (C, 1) is the type of the field;
Field_len (N, 3) is the length of the field;
Field_dec (N, 3) is the number of digits after the decimal point
(if the field is NUMERIC)
Field_recv (C, 80) is the caption
Field_dom (C, 80) is the field definition domain

2.3. The specialized library of classes

The use of a custom class library provides efficient application development tools which permit a reduction in software development time, as well as ensuring the high quality of the product. In this section, we present some important new classes that constitute the basis for the construction of screen forms. The pictures below are an illustration of the above mentioned classes.

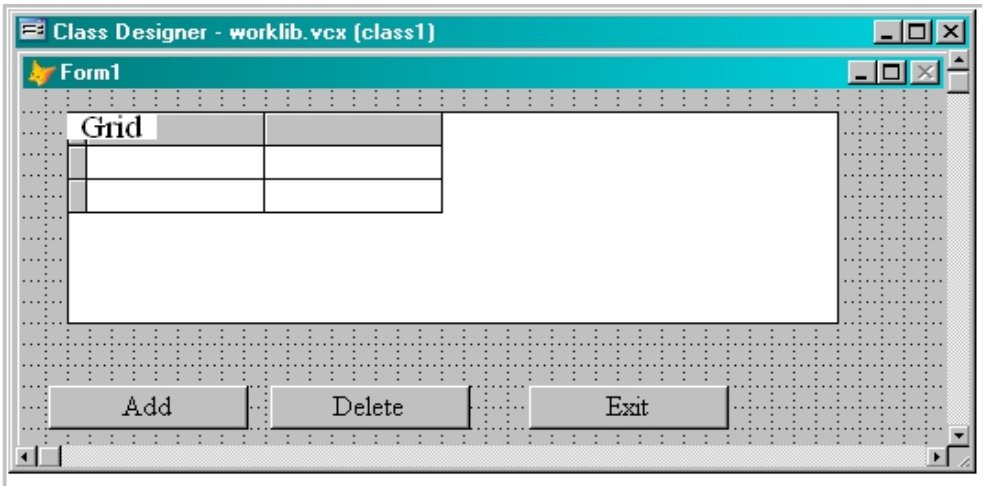

Figure 1. Class for the addition, deletion or correction of a record in a table.

The class in Figure 1 consists of an object grid which displays data from a table, and command buttons to perform corresponding operations on the table.

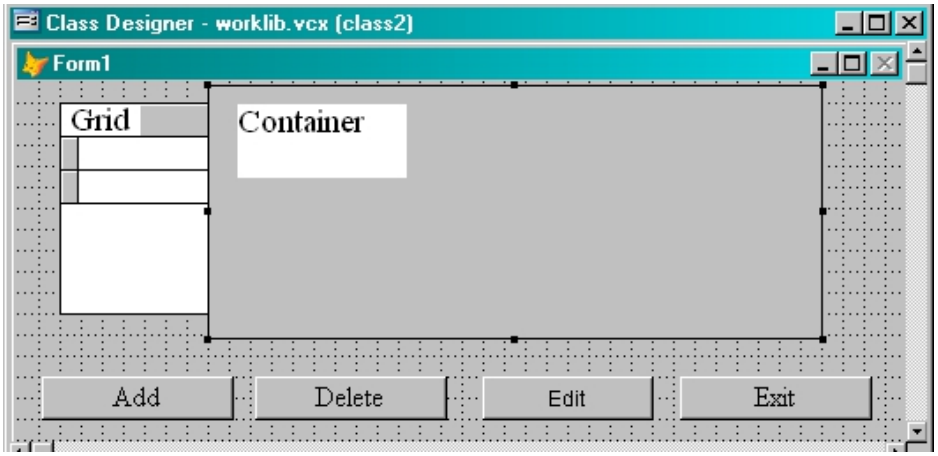

Fig. 2. Class for the addition, deletion or correction of a record in a table using the correction container.

In addition to all the components of the class in Fig.1, this class has the "Edit" command button which activates an editing mode using a special container to record modifications. After running a screen form based on this class, it initially displays only the "Grid" and all buttons. The "Container" becomes invisible. In this "browse mode" no changes of data are allowed. After clicking the edit button, the screen form switches to edit mode, allowing editing operations. In this mode the screen form displays only the container and all buttons. The grid becomes invisible. The caption on the "edit" button is replaced with "browse". If this is pressed again, "browse mode" is resumed.

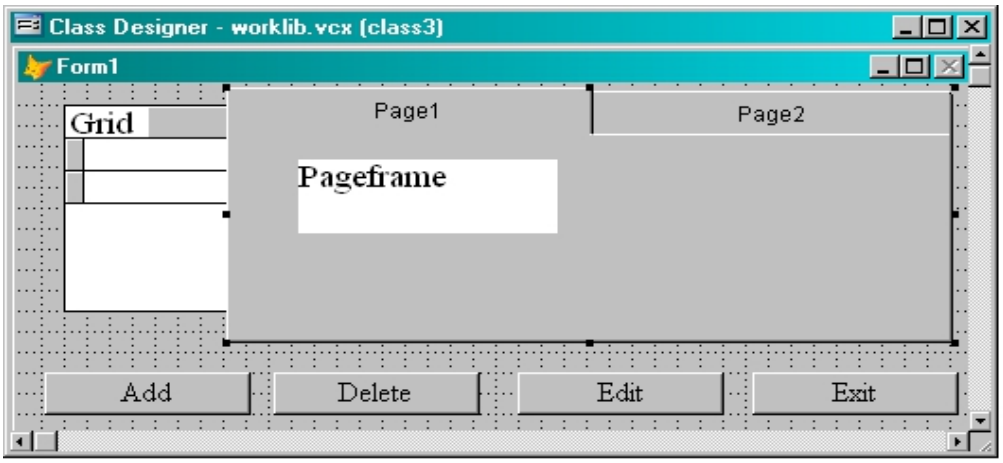

Figure 3. Class for the addition, deletion or correction of a record in a table using a multi-page representation (Page Frame).

The class in Figure 3 is used when the screen does not have enough space to display all the required fields of the table. Multi-page presentation is provided with the help of the "Page Frame" object. This class differs from the class in Figure 2 as the container is replaced by the page frame. The proposed list above is far from covering the entire set of classes that could be used for information-calculating applications. Among other possible classes are those for filtering tables based on the respective conditions set by the user, cursor - classes for the hierarchical representation of the database, classes that are based on combinations of existing ones etc.

Conclusion

This paper presents an approach to the creation of specialized technology for object-oriented applications based on the development of specialized tools, which include a formula interpreter, a form generator and a specialized library of classes. The implementation of all these components was carried out on the basis of the Visual FoxPro database system and has been demonstrated in practice in a series of commercial applications concerning computer aided accountancy and business correspondence.

References:

I. Jacobson. Object-Oriented Software Engineering - A Use Case Driven Approach. Addison Wesley, 1994.

Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. - ISBN 0-201-63442-2 Hardback, 416 p., 1995. Anton Eliëns, Principles of object-oriented software development, Addison-Wesley, 2000. DeLoach S.A., Hartrum T.C. A theory-based representation for object oriented domain models // IEEE Transactions on Software Engineering, Vol.26, No. 6 -2000 - P. 500-517.

Menachem Bazian, Visual FoxPro 6 - Edicion Especial, ISBN:970170343X, Prentice Hall 2000.