Xian

OBJECT-ORIENTED TECHNOLOGY

Proceedings of the 36th International Conference on Technology of Object-Oriented Language and System(TOOLS Asia'2000)&the 4th National Conference on Object-Oriented Technology and Application(OOT China'2000)

Edited by
Chen Ping
Software Engineering Institute, Xidian University

Xi'an,China
October 31-November 3, 2000

OBJECT-ORIENTED TECHNOLOGY

Proceedings of the 36th International Conference on Technology of Object-Oriented Language and System(TOOLS Asia'2000) & the 4th National Conference on Object-Oriented Technology and Application(OOT China'2000)
(October 31-November 3, 2000)

Edited by
Chen Ping
Software Engineering Institute, Xidian University

Sponsors

National Natural Science Foundation
Xi'an Municipal Science and Technology Commission
The Administrative Committee of Xi'an National Hi-Tech Industrial Development Zone
Interactive Software Engineering (ISE) Inc. , USA
Development Center of Xi'an Software Park, China
Xidian University
School of Computer and Information Science, University of South Australia
Department of Computer Science and Technology, Tsinghua University
State Key Lab. for Novel Software Technology, Nanjing University
Department of Computer Science and Technology, Xi'an Jiaotong University
Department of Computer Science and Engineering, Northwestern Polytechnical University
Department of Computer Science and Engineering, Northwest University
Software Engineering Institute of Xidian University

Preface

The International Conference on Technology of Object-Oriented Languages and Systems (TOOLS) is a serial international conference on object-oriented technology and applications. It is held three times each year in Australia, Europe and America separately. In 1997, it first moved to Asia and was successfully held simultaneously with the 1st Conference on Object-Oriented Technology and Application (OOT), which was the first national conference on object-oriented technology, at Beijing in 1997. The great successes of these two conferences aroused strong enthusiasm of the participants. Since then, these two conferences have been held in China each year. As the successor of them, TOOLS Asia'2000 and OOT China'2000 will be held at Xian New Century Hotel in October 2000. It is going to be another academic meeting which provide the scholars from both China and other countries with the opportunity to exchange widely their research achievements. Scholars in this field from all over China responded actively to the calls for papers to these two conferences. After three rounds of careful examination and the final approval of the Programming Committee, about one third of the paper are rejected. Unfortunately, TOOLS Asia'2000 Proceedings still can not include all good papers because of the limited space. As a remedial measure, we, as usual, decided to edit another proceedings entitled "TOOLS Asia'2000 and OOT China'2000", which include both the English articles that are not included in TOOLS Asia'2000 Proceedings and the Chinese ones accepted by OOT China'2000. The authors of these papers will enjoy the same right of presenting their papers at the two conferences. The proceedings consist of 6 sessions:

Object-Oriented Languages aud Systems, Object-Oriented Analysis and Modeling, Object-Oriented Software Development, Distributed and Concurrent Systems, Database Systems and Applications, Applications and Experiences.

The conference will be held in Xi'an, a historic place in China, at the most pleasant season. Xi'an is a city with a mixture of ancient culture and modern civilization, famous in its very long history. It was the starting point of the well-known Silk-road in the ancient time and will serve as an entry to accommodate new technologies in the new century. Xi'an is also the portal of the great west of China, which is now the focus of the whole country. Because of the siguificance of the plan of "Developing the Great West" in booming China's economy and of the enormous opportunity for foreign investment, Xi'an is now

attracting the attention of both domestic and international organizations. This background makes the Conference particularly important for both local and international research and industrial communities.

I would like to thank the sponsors very much, they provided the success basis of the conference. I also want to thank the Programming Committee, Organizing Committee and especially the Secretariat of these two conference for the great efforts they have made and the arduous work they have done, which ensured the success of the two conference. Finally, I thank all the authors for their participation with great enthusiasm.

Prof. Ping Chen

Table of Contents

Distributed Systems and Parallel Systems

Agents

Database

Components

Web and Internet

A METHOD FOR BINDING ADA95 GENERIC TO C++ TEMPLATE

Liu Jian Yu Haibo
(Software Engineering Institute Xidian University Xi'an P.R.China 710071)
E-mail: liujian@sei.xidian.edu.cn

ABSTRACT

This paper proposes a "preprocessor" for binding Ada95 generic to C++ template. The main idea is taking the instantiations of generic and template from compilers, generating the instances by preprocessor, and therefore changing the template binding into relatively compiler independent class binding.

Keywords: OOPLs, Bindings, Ada95 generic, C++ template, preprocessor

1 BINDINGS AND BINDING TOOLS

Ada95 is the first and only internationally standardized object-oriented programming language (OOPL). In Ada95 programming environment, other language resources are often reused by bindings[1]. The basic idea of binding is providing user the Ada95 interface of the other language resource, and in the implementation it is linked to the binary code of the resource.

C++ is one of the most popular OOPLs with plenty of reusable classes. Since only procedural based C, FORTRAN, and COBOL binding interfaces are provided in Ada95[2], it is important to study binding technology based on classes of OOPLs.

For this reason, some companies provide C++ class binding products in Ada95 Environment, for example, MFC bindings in ObjectAda[3]. But it is not enough only to have binding products, a binding tool is also needed to build bindings on any up to date and well verified C++ application classes.

2 THE BINDING TOOL FOR ADA95-C++ CLASS

The process of binding C++ class is shown in Fig. 1. XDBinds (shadowed block in Fig. 1) is A binding tool[4] which does three principal works.

<1> Analyses C++ class specification (*.h file after C++ preprocess), fills the

information into a symbol table according to the mapping relationships between C++ class and Ada95 package;

<2> Analyses C++ target code (*.obj, *.lib), extracts binding information (binary formed C++ link-name) and adds it into the symbol table;

<3> Generates Ada95 class package (both specification and body) from the information and attributes of the symbol table.

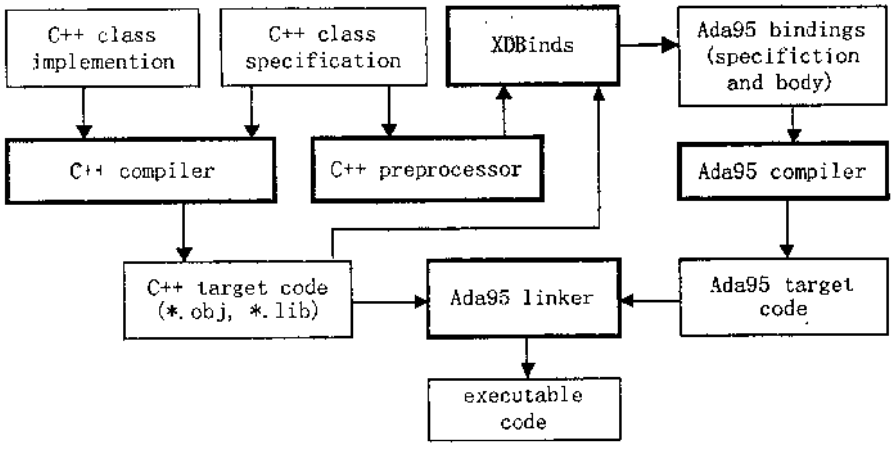

Fig. 1 The process of bindings

3 BINDING ADA95 GENERIC TO C++ TEMPLATE

3.1 Difficulties in Generic-Template Binding generation

Both C++ and Ada95 support parametric polymorpnism. C++ provides template to describe a group of classes with similar behaviors. The C++ template correspondence of Ada95 is generic package (generic for abbreviation). The function of both template and generic is to leave the similar and repeated definitions to the compiler, one may define a single template or generic source, and at instantiation time, the compiler copies the template or generic source with replacing parameters by arguments to generate an instance. Therefore template and generic are both based on source reuse but not target code reuse.

The binding from Ada95 to C++ is the reuse of binary code, it is through Ada95 compiler pragma instruction and link-name to realize the call between different languages. Since C++ template is the source reuse, compiler does not generate the target code before the instantiation of arguments. Meanwhile, Ada95 generic package body needs link the implementation of methods and attributes to C++ target code. Therefor, one cannot get available generic package before the template instantiation. That means, to get available binding package, the template must be instantiated before the instance is compiled.

The difficulty in template binding is that Ada95 generic and C++ template is instantiated at compile time. It is unfeasible at Ada95 generic instantiation time to notify C++ compiler instantiating the template, generate the C++ target code witch Ada95 methods and attributes need, and then abstract link-name from target to fill into the corresponding pragmas. The reason is that the compilers are not open enough. Ada95 compiler can not stop generic instantiating to wait C++ compiler's template instantiation, and the binding tool is not allowed to intercept the information to notify C++ compiler at generic instantiation time. The binding tool can not dynamically interact with Ada95 and C++ compilers at compile time.

3.2 A Solution for Generic-Template Binding generation

To solve the problem above, a "preprocessor" of separating the template instantiation from the compiler and combining the preprocessor into the binding tool is proposed. The solution can be obtained in the following four steps:

1. The mapping of C++ template to Ada95 generic is added into the XDBinds to generate Ada95 generic package specification for C++ template, if necessary. At the current stage, Ada95 generic package body can not be generated yet.
2. If users programming with the generated generic package specification, that is, there are instantiation statements and references to the instance in user's packages (shown as the dashed line block in Fig. 2), the template binding preprocessor should be used to generate instance packages at step 3. Before step 3, user's packages could be compiled to make sure there is no syntax errors in the program.
3. The binding preprocessor is used to do the instantiation for compilers. For Ada95 application package, the generic instantiation statements and the references to generic resource are taken off from the package, and the references to an "instance" package resource are used instead. For C++ template, the C++ template instantiation of Ada95 instantiations' arguments is made, and an "instance" class is generated, then the class is compiled to generate the binary code.
4. Again, the instance class specifications and the binary code are taken as the input, the Ada95 package with the same name as the Ada95 generic package is generated and added into user's files to form the complete application programs.

Above steps are shown in Fig. 2, the correspondences of steps 1, 2, 3 and 4 are marked in Fig. 2 as ①, ②, ③ and ④. XDBinds is used twice: first time it is used to bind C++ template to generate the generic package specification; second time it is used to bind instance C++ classes which generated by the preprocessor.

The solution keeps the method consistency with template binding and class binding. In the solution, the Ada95 generic package is replaced by the instance package with the same name. Since the package is instantiated from the generic package, the interface of the attributes and methods are kept unchanged, so there is no effect on the references of the attributes and methods after the preprocess.

3.3 The Validation of Generic-Template Binding

In order to validate the feasibility of the proposal, a simple C++ stack template class is given, and the steps of how the corresponding Ada95 packages can be generated are shown.

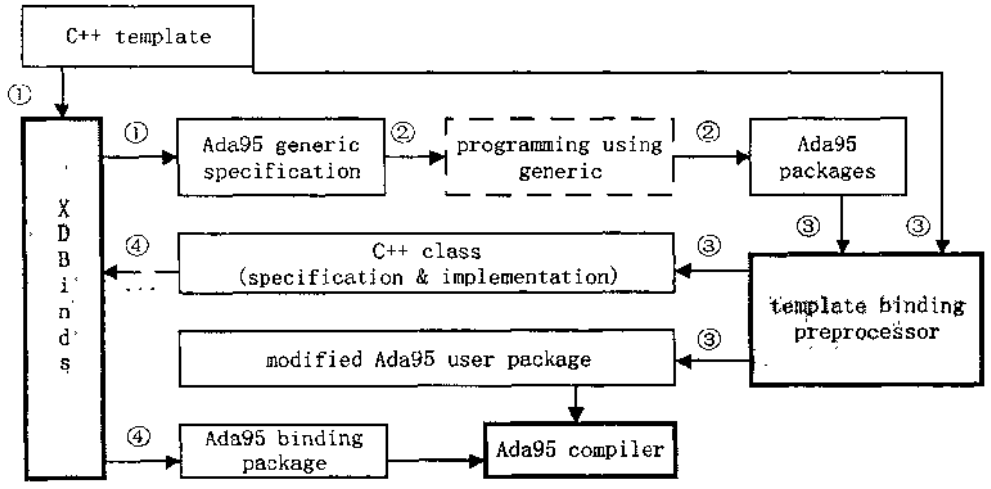

Fig. 2 The process of C++ template bindings

3.3.1 The specification of C++ Template

C++ stack template specification is as follows:

```
template<class T, int N>
class TStack{
    public:
        TStack(); // construct function, set SP to -1.
        void Push(T &e);
        T Pop();
        T Top();
    private:
        T Item[N]; // use array to implement stack
        int SP;    // stack pointer
};
```

Normally, the specification and implementation of the template are required in the same file by C++ compiler. Since only the specification is concerned here, the implementation is omitted. The corresponding Ada95 generic package specification is generated by XDBinds as follows:

```
generic
    type T is private;
    MAX_STACK : in Positive := 3;
```

```
                -- the default value for MAX_STACK, it may be changed later
    package Class_stack is
        type Stack is tagged private;
        procedure push(the:in out Stack; item: in T);
        procedure Pop(the: in out Stack);
        procedure Top(the: in out Stack; item: out T);
        procedure initialize (the : in out Stack);
    private
        type Stack_array is array (1..MAX_STACK) of T;
        type Stack is tagged record
            elements: Stack_array;
            tos    : Integer :=0;
        end record;
    end Class_stack;
```

3.3.2 The reference of generic package

After the generation of the Ada95 generic package, users can use the attributes and methods of the package to write their application programs. For example:

```
    with ada.text_io;  with Class_stack;
    procedure main is
        package int_io is new ada.text_io.Integer_io(Integer);
        use int_io;
        package Stack_int is new Class_stack(Integer,20);
                    -- instantiation of the generic package
        use Stack_int;
        number:Integer;
        number_stack: Stack;  -- declares a stack object
    begin
        get(number);
        push(number_stack, number);  -- use the method of the instance
    end main;
```

Meanwhile, since there is no implementation body for the generic package, the users' application program can not be compiled and linked to the executable code. The only thing that compiler can do is to find the syntax errors of the application program.

3.3.3 Preprocesses of Generic and Template

After making sure the program has no syntax errors, preprocesses of generic and template can be made then. The instantiation statement is removed from the generic package, and the replacement of a reference for the instance package with the same name is made as follows (even through the package has not been generated yet):

```
    with ada.text_io;
```

```
with Stack_int; use Stack_int;
procedure main is
    package int_io is new ada.text_io.Integer_io(Integer);
    use int_io;
    number:Integer;
    number_stack: Stack;
begin
    get(number);
    push(number_stack, number);
end main;
```

The above replacement of the package does not change the references of the methods. According to the arguments and variables of the instance of the Ada95 application program, an instance class of the C++ template is generated as follows:

```
class TStack{
    public:
        TStack(); // constructor, sets SP to -1
        void Push(int &e);
        int Pop();
        int Top();
    private:
        int Item[10];  // array as a stack
        int SP;        //  pointer for top of the stack
};
```

As mentioned above, the specification and implementation of the template are required in the same file. This is the key requirement of the instantiation, only this can the parameters be replaced with arguments both in specification and implementation to generate the complete class definition. For same reason, the implementation is also omitted.

The generated instance class is saved as the C++ source file, and then compiled to get the target code file.

3.3.4 The Binding generation

After the preprocess, XDBinds is used to generate the corresponding Ada95 binding package with the same name as the generic substitution obtained previously as follows:

```
with VC;  -- provided by XDBinds
package Class_stack is
    type Stack is tagged private;
    procedure push(the: in out Stack; item: in VC.int);
    procedure Pop(the: in out Stack);
    procedure Top(the: in out Stack; item: out VC.int);
```

```
    procedure initialize (the : in out Stack);
private
    type Stack_array is array (1..MAX_STACK) of VC.int;
    type Stack is tagged record
        elements: Stack_array;
        tos     : VC.int :=0;
    end record;
end Class_stack;
```

Up to now, binding Ada95 generic instance to C++ template instance is achieved.

4 CONCLUSION

Facing to the difficulties of the Generic-template binding generation, this paper proposed the "preprocessor" solutions. First, C++ template instantiation is translated into Ada95 instance package specification and the corresponding C++ instance class is generated. Second, Ada95 binding from the C++ instance class is generated. The are two advantages of the solution, (a) Template instantiation and the class binding are separated into two separate parts, the template binding, template instantiation, and class binding are combined into the unified scheme. If there is no instantiation in users application packages, the binding process is as the same as class binding. (b) The problem that binding tool can not dynamically interact with compilers is got rounded, so that the binding tool and compilers are working in loosely coupling mode, this provides the foundation for constructing the compiler independent binding tools.

The feasibility of the solution is verified, but the implementation of the solution is left open, there is still a lot of work to do for template binding preprocessor implementation.

REFERENCES

[1] Defense Information System Agency, Ada95 Bindings Report, Aug. 1995, http://www.adaic.org/AdaIC/docs/bindings_report

[2] International Standard, ANSI/ISO/IEC-8652:1995 Ada95 Reference Manual(Annex B, Interface to Other Languages), Jan. 1995

[3] Anoix Inc., Ada Foundation Classes User Guide, 1998

[4] Yu Haibo, Ada95-C++ Binding Technology Study, master thesis, Xidian University, Xi'an, P.R.China, 1999

Object Oriented Method in Geotechnical Engineering

——An Object Oriented Finite Element Analysis System

Xiang Yang, Ge Xiurun
(Institute of Rock and Soil Mechanics, Chinese Academy of Sciences
Wuhan, Hubei, P. R. China, 430071)

ABSTRACT

This paper presents the application of object oriented technology in geotechnical engineering. The analysis and implementation of an object oriented finite element system is discussed. It is that object-oriented programming can be used to significantly improve finite element analysis programs. It provides better program performance.

Keywords: Object-Oriented, software reuse, software architecture

INTRODUCTION

Geotechnical engineering is a branch of the civil engineering. The geotechnical engineering software is used to analysis the stability of the foundation of the building, the slope, the tunnel and so on. Most of the software use finite element method (FEM), which applied in many other fields such as acoustics, nuclear physics, heat transfer and fluid mechanics. This paper describes the problems associated with the conventional geotechnical engineering FEM software and the solution offered by the object-oriented technology. And an objected-oriented FEM analysis system based on Windows is illustrated.

BACKGROUND

Nowadays, computer is widely used in civil engineering. Most of the software use finite element method. In this area, finite element method has been developed for about several decades. Almost all of the finite element problems include the following main steps:

1)Preparation of the data for analysis which include model developing and mesh generating;

2)Data input;

3)Calculation of stiffness matrices and load vectors;

4)Assembly of global stiffness and load vectors;

5) Application of constraints;

6) Solution of equations;

7) Result output and analysis, which may include some graphical processing.

The exact order and the form in which these steps occur may vary from program to program, but they form the basis for all finite element programs. The first step could be named pre-processing while the last step post-processing.

This method was first used in the 1950's and its use increase rapidly in the following years. Up until a few years ago it was implemented mainly on mainframe computers. With the rapid development of the personal computers, the FEM programs are becoming increasingly available on personal computers.

PROBLEM

Coupled with the development of hardware has been the development of software and programming languages. The finite element method has conventionally been programmed in Fortran that is a procedure-oriented language. A procedure-oriented finite element program may cost years of developing, yet the reusability of the program is not satisfiable. When a new algorithm or a new kind of element is designed it is difficult to apply it in the software because the alternation of one subroutine may affect the whole program. The risk of the error generation may be increased greatly. The reusability is the important norm to judge the performance of the software. It includes the reuse of the concepts, the documents, codes and so on. It is the combination of technology, method and procedure.

What's more, the pre-processing and the post-processing are the bottleneck of the application of the FEM technology. The automatic mesh generation is the painful task troubling software engineers and civil engineers. And the work of pre-process is very fussy. Only a select group of individuals has the technical expertise required to apply this method in professional practice. A strong effort is being made by developers to supply analysis programs as black box modules to be used with CAD system. Conventional programs are not easily reshaped to perform the desired tasks, so developers must redesign and rebuild their finite element libraries.

With lots of merits, the object-oriented technology reflects the concerns of software developers. The object-oriented approach is very attractive for many applications in finite element analysis, especially in geotechnical engineering.

OBJECT ORIENTED ANALYSIS

Object oriented method includes analyzing the object, estahlishing the layer structures of the classes and developing the system frame. The iuethod means the description of the classes, the consistency between the problem and the solution. The common main steps are listed as follow:

1) Analyzing the system and deciding the object. The problem considered should be the user

interface, the information object, the routine management object and the system object.

2)Establishing the analyzing models and the interaction between the models.

By the guidance of the thought, an object oriented finite element analysis system is developed which we give its name MEG. It consists of three main parts: the pre-processor, the finite element analyzer and the post-processor. The pre-processor in fact is some kind of CAD system, which is used to model. But it is not only the CAD system, but also the mesh generator that has the self-adaptation ability.

The software architecture of MEG has the object-oriented style which based on data abstraction and object-oriented organization. Data representations and their associated primitive operations are encapsulated in an abstract data type. The components of the style are the objects. The organization structure is shown in Fig. 1.

The system consists of six modules. The user interface links the user and the system. The user establishes the graphical objective model for finite element analysis and checks the result of the calculation. The database management module exchanges the data from the front part to the process part. The core of the system is made up of the pre-processor, the analyzing core and the post-processor. After the mesh generation of the pre-processor, the database manager sent the information of the mesh to the user interface. Then the user validates it and sends message to the calculation core. The post-processor gets the information of results and shows it to the user. The modules communicate by dispatching message. When the object receive message it reacts accordingly. Different objects will react differently when they receive the same message.

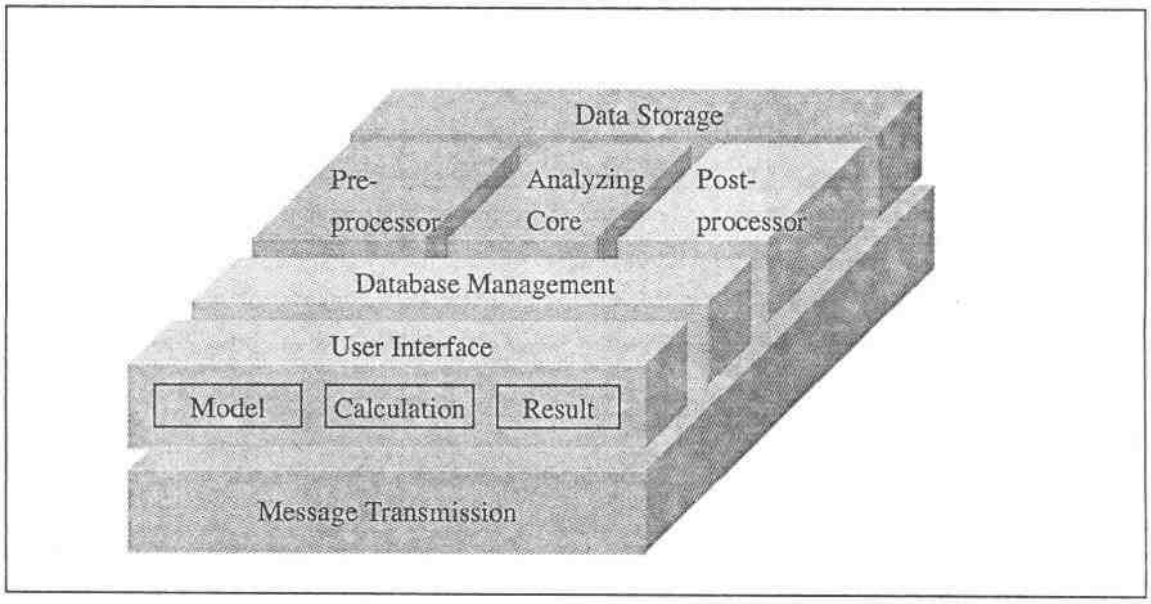

Fig. 1 The organization structure of MEG