第6卷第1期
1995年1月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 6, No. 1
Jan. 1995

基于可重用方法的库与应用程序接口开发环境*

杨家海　柳西玲

(清华大学计算机系,北京 100084)

摘要　当前,开发以数据库为核心的集成化CAD系统已迫在眉睫,为支持这种集成化,提供一个能自动生成数据库和应用程序之间接口的开发环境是很重要的手段之一.本文在分析了这种接口程序的功能和一般结构以后,提出了一个基于可重用方法的库与应用程序接口开发环境的总体模型——DDBUIS,试图结合两种支持复用的方法自动生成库与应用程序之间的接口,并为软件开发人员在开发其他系统时提供可重用的支持.

关键词　用户接口,软件复用(重用),程序自动生成.

计算机辅助设计系统是一个由众多应用软件组成的综合系统.以前各种应用软件是独立开发,自成体系的.随着CAD技术的飞速发展,开发以数据库为核心的集成化CAD系统已迫在眉睫[1].一个集成化系统要求各应用程序有统一的用户观点,统一的操作模型和统一的数据存放方式.统一的用户观点要求系统中各应用程序有一致性的用户界面,为实现统一的数据存放方式,可通过设计一个集中性的数据库来实现.各应用程序间的数据传递都通过数据库实现.

然而各应用程序使用各自的数据,其结构、格式也各不相同,更重要的是各应用程序内存数据结构与库可存储的数据格式不同.所以,对这类集成化系统来说,每个应用程序与库之间都必需有一个接口程序去实现数据格式间的转换[2,3].以往,这种接口程序都是在应用程序与库集成时手工编制的.对于设计步骤繁多的工程设计自动化系统来说,这部分工作量也是相当可观的,更为严重的是,应用程序及其数据结构的微小改动也可能导致整个接口程序重新编制.

但是,如果对这种接口进行细致的分析解剖,并抽象到功能角度看,我们发现,虽然它们表面上有很大差异,但本质上有许多相似之处,一旦数据库的物理存储模式确定,接口不外乎是通过一个缓冲器与库进行读或写的操作,在读时,是将库内批量数据分离成应用程序所需的某种格式,而在写时,是将分散的应用程序数据格式集中成库批量的物理存储格式,这

* 本文1992－04－03收到,1992－09－07定稿

作者**杨家海**,1966年生,助教,主要研究领域为软件,CAD数据库,网络,协议一致性测试.**柳西玲**,女,1936年生,副教授,主要研究领域为软件,CAD数据库及应用软件.

本文通讯联系人:杨家海,北京 100084,清华大学计算机系

种分离和集中的过程实质上是对数据进行分类组织转换的过程.由此抽象的分析结果启示我们去开发一种能自动生成接口程序的工具.它们将集成化系统的应用程序和库的界面统一,并且自动生成接口,以减少手工编制接口的高成本和许多繁琐的、重复的开发工作,从而使应用程序员摆脱与设计库之间频繁交换数据的细节问题,提高系统开发速度[4].

此外,一个开放的集成化系统要求能不断增加应用程序,接口的自动生成对集成化系统的维护开发、不断扩充也有重大意义.

为此,本文提出了一个基于可重用方法的库与应用程序接口开发环境的总体模型——DDBUIS,试图结合两种支持复用的方法自动生成库与应用程序之间的接口,并为软件开发人员在开发其他系统时提供可重用的支持.

1　基于复用的软件开发方法

软件技术的发展经历过几次波折,每次波折过后人们就开始反思原有的一套技术,并开始探索新的技术.在对新的软件开发环境的研究中,软件重用思想受到了人们的普遍重视,并正被人们当作解决软件生产率和提高质量问题的关键而加以研究.因为研究结果表明,在存在着生产和需求的巨大矛盾的同时,软件开发过程中还存在着大量的重复劳动.人们希望在新的软件开发环境中能利用以前开发中的工作,减少重复开发的劳动量[5-7].

目前,用于支持软件复用思想的方法主要有两种:块式复用,即使用可重用的程序块;模式复用,即通过程序转换获得可重用性.

基于块式复用的软件开发方法是以可重用成分库为数据基进行的,可重用成分经过确认和描述后被存放在可重用成分库中,从库中检索出的成分经理解和修改后被复用,可重用成分库是复用过程中的核心部分.为支持这种方法,开发系统必须具有支持可重用分量的设计、存储和检索能力,并具有将现有的多个可重用分量组合成一个特定分量的机制.

模式复用是通过构造目标的结构,由生成器(程序转换机制)生成特定目标来获得可重用性的.一般地,开发系统具有一个有关特定域的可重用模式,基于这个模式和具体的领域信息生成目标程序.或者说,可重用模式是领域的模型,系统(生成器)接收用户对目标的抽象描述(亦即具体的领域知识)生成实例化的目标[8].为支持这种方法,系统必须具有模型化领域的能力(或具有领域模型)和提供给用户输入具体领域知识的能力.

下一节我们将首先讨论接口程序的功能和结构,并抽象出接口的一般模型作为可重用模式.

2　接口程序的功能、结构

从接口程序在集成化系统中的工作流程图(图 1)可以看出:接口程序由两部分组成,一是公用读写函数,另一个是转换程序.前者完成对设计库中数据的实际存取,独立于应用程序,而后者则是实现各应用程序内存数据格式和库可存储的数据格式的转换,依赖于应用程序.

如果从功能上分,则每个应用程序的接口程序都由这样两部分组成即接口读函数和接口写函数,前者把指定单元的指定版本的数据从设计库中取出,并将数据转换,分离成应用

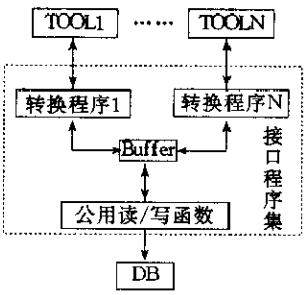

图1 接口程序工作流程图

程序所要求的某种格式提供给应用程序;而接口写函数则是将分散的应用程序的数据格式转换成库可存储的数据格式并存入库中适当的地方.

一般地,任何一个接口程序都具有如图 1 所示的结构,不同的应用程序,对应不同的接口程序有不同的转换程序,但仅仅是转换的内容不同,其转换的一般格式还是不变的.

剖析转换函数的一般结构,可以发现,每个函数都不外乎由这样几个部分组成,即:函数头、函数形式参数说明、局部变量定义、函数体等,而函数体则由顺序语句序列和一个循环结构构成.我们可以用一种高度抽象的语言来分别表示各个语法组成单位.有了转换函数的各语法单位的抽象表示,我们可得到转换函数的抽象表示如下:

```
function header ((parameter table))
    〈parameter declaration〉
    BEGIN
    〈local variable definition〉
    〈sequence of statements〉
    WHILE (pointer IS NOT NULL )
        LOOP
            〈sequence of transition statements〉
    END LOOP
    〈the remainder of sequence of statements〉
END
```

这是基于模式复用生成接口的基础.

3 DDBUIS 的接口描述语言

用户使用 DDBUIS 生成其接口程序时,首先要用接口描述语言将其接口要求描述出来,也就是说,基于模式复用生成接口的系统必须要有提供给用户输入具体领域知识的能力.

在人工编制此类数据库与应用程序之间的接口程序的过程中,我们发现,只要知道了对应用程序的全部数据结构的描述,即可编写接口程序.通常,基于结构化程序设计语言的应用程序,其数据都能以结构为单位进行组织、存取(最极端的情况,无非是结构中只有一个数据项,但这是允许的).所以,如果用结构对应用程序的数据结构进行描述,则这种描述是完全的.考虑到 C 程序设计语言目前已被广泛接受和使用,我们采用一种类似于 C 程序设计语言的定义语句的语言作为描述接口的工具.它也包含常量定义、简单类型定义、结构类型定义和联合类型定义,而且还包含了组织整个视图数据的视图结构定义,通过它可以描述应用程序的全部数据,从而也就描述了接口程序工作的全部内容.

由于应用程序的数据可以统一到结构的基础上来组织、存取,所以,对结构的描述是描述语言的核心.结构描述的形式(部分)如下:

```
〈结构〉::=typedef struct〈结构标志〉
        "{"
        〈域定义〉{〈域定义〉}
        "}"
        〈结构类型名〉;
〈域定义〉::=〈类型名〉〈变量名〉;[〈辅助信息〉]
〈类型名〉::=〈标准类型名〉|〈已定义类型名〉|struct〈结构标志〉
〈标准类型名〉::=char |double | ……
〈辅助信息〉::="%"〈信息〉"%"
……
```

接口描述语言为应用程序生成与数据库的接口提供了一种语言描述工具,应用程序的用户利用它把接口要求描述出来,这种接口要求包括了应用程序中需要存储的全部数据的数据结构的描述.一方面,应用程序据此组织它的数据;另一方面,系统将根据这个描述生成接口程序,接口程序把应用程序的数据结构转换成库可存储的数据模式,并实现数据的实际存取.

由于应用程序的数据包含了结构指针等各种动态数据,而数据库只存储静态的数据.因此,接口程序的这种数据格式的转换是整个接口程序实现的关键,而接口程序的生成是DDBUIS系统的主要功能.所以,接口描述语言的设计和实现是整个DDBUIS系统功能实现的基础.限于篇幅,本文未能给出接口描述及接口程序生成的有关例子.

4　DDBUIS的结构

DDBUIS的基本结构如图2所示.

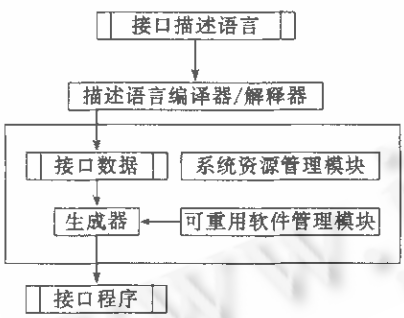

图2　DDBUIS系统结构示意图

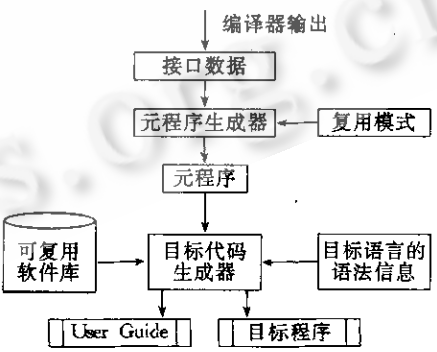

图3　生成器的结构

DDBUIS可分为四大部分:描述语言编译器DLC、接口程序生成器Generator、可重用软件管理模块RSBMS、系统资源管理模块Manager.

描述语言编译器将用户所写的接口描述语言转换为接口程序生成器所用的接口数据.用户使用DDBUIS生成其接口程序时,首先要用接口描述语言将其接口要求描述出来;接口程序生成器是DDBUIS的主要模块,它根据用户的接口描述产生相应的接口程序.同时,生成一些帮助用户理解接口程序的注释信息.此外,由于生成的接口程序将直接对库操作,所以,该部分还必须考虑并发机制;可重用子程序管理模块是DDBUIS的另一个重要部分,

作为DDBUIS的一部分,它为接口的生成等提供可重用服务.另外,RSBMS可以作为一个相对独立的子系统为用户在其他软件的开发中提供可重用服务;系统资源管理模块包括应用程序的登录、维护,接口程序的管理维护等,为集成系统的用户提供一些查询系统资源的服务程序,以帮助他们了解系统的集成情况.

关于可重用软件管理部分的设计及实现已在另文作了论述,限于篇幅,下面我们将主要讨论接口程序生成器的设计及实现.

4.1 接口程序生成器

接口程序生成器是一类程序生成器,它是DDBUIS的核心.接口程序生成器在接收由编译器生成的接口数据以后,就根据接口的复用模式生成接口程序的抽象描述——我们称之为元程序(Meta—Program).或者说,我们对接口的可复用模式用一种更形式化的方法表示.此后,由接口数据驱动生成器,并根据目标语言的语法信息把元程序转换成可执行的目标代码.与此同时,也生成一些帮助用户使用目标程序的用户指南(User Guide).

生成器的基本结构如图3.

从图3可以看出,接口的生成与复用有密切关系,或者说,接口的生成是一系列复用和转换的结果.第2节,我们已经讨论过接口程序的一般结构,并且指出接口程序的所有可变特性都反映在其转换函数部分.我们还给出了转换函数的抽象描述.接口生成时,元程序生成器将根据接收到的接口数据生成一系列的具有类似该模式的转换函数的抽象描述——元程序(Meta—Program).我们称之为元程序是因为它和任何具体的语言无关,但又可以转换为具体语言的可执行代码.

之所以要采用这种抽象的描述方式,主要是考虑到多目标性.事实上,如果一开始就用某种程序设计语言的语法结构来表示,也是可以的,但这样一来,生成的接口程序就只能是以该语言为目标语言了.相反地,采用这种抽象的描述性的方式作为过渡,则在后一阶段,可根据不同语言的语法知识把接口的抽象描述转换成具体目标语言的可执行代码.

有了元程序和接口数据,再结合具体的目标语言的语法,可以很容易地生成接口转换程序.目标代码生成器根据元程序选择合适的(一般情况下是一一对应的)目标语言的语法单位以最终生成目标代码.在此过程中,根据功能需要自动选择软件库中的子程序,嵌入到生成的接口程序中.

公用读写函数是一类可重用的子程序,在接口程序的生成中,它们独立于具体的应用程序,所以,它们编制完成以后,可以先经过调试,放入库中,在接口生成时随时取用.

4.2 DDBUIS与应用程序的连接

DDBUIS是一种辅助软件开发环境,目前它的主要功能是为用户快速生成应用程序和库之间的接口程序.应用程序使用各种辅助工具时最常用、也是为大多数编程人员所熟悉的方法是调用子程序.针对这个特点和DDBUIS的功能要求,系统被设计成独立运行的工具.即DDBUIS本身不集成入整个集成化CAD系统.DDBUIS运行完成以函数的形式向应用程序提供它所需要的功能.

用户使用DDBUIS生成其接口程序时,首先要用接口描述语言把接口要求描述出来,然后再运行DDBUIS输入接口描述生成接口程序.在生成接口程序的同时还生成使用某个接口程序的帮助信息,诸如接口程序的调用格式、参数要求等.查阅这些信息可以知道应用

程序如何与接口程序连接.

以下是应用程序调用接口程序的一般格式:

```
Application( )
    {
    variable_type variable;
      ……
    r_interface(variable);
      /* 从库中调出上次设计的结果数据 */
      ……
    w_interface(variable);
      /* 将本次设计结果数据写入库中 */
    }
```

这里 variable_type,按目前的实现(以视图为单位组织数据)是视图数据结构类型名,variable 则是视图变量.

设计开始时,应用程序首先调用接口读函数,把上次对某个单元的设计结果读出来,在此基础上开始设计.本次设计完成后,或者作为最终结果,或者作为下次设计的依据,把设计数据写入库中.一般来说,一次设计后的结果总要保存下来以备后用,所以接口写函数在每个应用程序中总是需要的,而接口读函数则不一定非要不可,因为用户在设计一个新的单元时是不会有该单元以前的数据的.

应该说,这种连接方式不管系统实现如何,但对应用程序来说还是很简洁、自然的,这是我们的目标之一.

5　结束语

目前,DDBUIS 的基本框架已经在 SUN4/110的 UNIX 操作系统上实现,从接收用户对接口的描述经过编译(生成接口数据)到接口程序的自动生成,系统已经可以部分地投入使用(用于自动生成接口),经过测试效果良好.

集成化是一种趋势,当前,开发以数据库为核心的集成化 CAD 系统更是紧迫,为支持这种集成化,需要做很多工作,其中提供一个方便的辅助软件开发环境就是很重要的一项工作.本文是对这方面工作的一个尝试,我们已经看到这项工作的重大意义和它的巨大的发展前景.

本文仅仅是一个开端,所需做的工作还很多,今后我们将进一步开发系统实现目标中未实现的功能,并进一步完善现有的功能.

参考文献

1　Edmundlien Y, Liu Xiling, Hong Xianlong *et al*. PANDA——an integrated VLSI CAD system. Proc. of the 2nd International Conf. on Solid State and Integrated Circuit Technology, Oct. 1989.

2　Li W H, Switzer H. A unified data exchange environment based on EDIF. The 26th ACM/IEEE Design Automation Conference, 1989.

3　Ernst Siepmann. A data management interface as part of the framework of an integrated VLSI—Design system. ICCAD Digest of Technical Papers, 1989.

4 杨家海,柳西玲. CAD工程数据库与应用程序之间接口管理系统. 第六届全国IC—CAD学术会议论文集.
5 Neighbors J M. The draco approach to constructing software from reusable components. IEEE Trans. on Softw. Eng., 1984, SE—10(5).
6 Burton B A, Aragon R W, Bailey S A *et al*. The reusable software library. IEEE Softw., 1987, 25—33.
7 Cheatham T E. Reusability through program transformations, IEEE Trans. on Softw. Eng., 1984, SE—10(5).
8 Luker P A, Burns A. Program generators and generation software. Journal of Computer, 1986, 29(4).

REUSABILITY—BASED INTERFACE DEVELOPING ENVIRONMENT BETWEEN DATABASE AND APPLICATIONS

Yang Jiahai　Liu Xiling

(*Department of Computer Science, Tsinghua University, Beijing* 100084)

Abstract　Presently, it's more urgent to develop the integrated CAD system that the database is the kernel. In order to support this integration, many jobs need to be done, and all of them, it's a very important means to provide a convenient developing environment which can generate interface between database and applications automatically. In this thesis, they analyzed the general function and structure of this kind of interface, and then, put forward a general model of interface developing environment between database and applications, which based on software reusable method—— DDBUIS. Combining the two reusable methods, DDBUIS attempts to generate the interface between database and applications automatically, and also provides reusable supporting for software developers during developing other systems.

Key words　User interface, software reusability, program generation automatically.