模型库管理系统的设计和实现*

李 京[1,2] 孙颖博[1,2] 刘智深[1,2] 张道一[3]

[1](中国科学技术大学 合肥 230026)

[2](中国科学院软件研究所计算机科学开放实验室 北京 100080)

[3](香港城市大学 香港)

E-mail: lij@cs.ustc.edu.cn

摘要 模型库管理系统是决策支持系统的核心,其基本功能包括模型的定义、组合、存储和调用.该文介绍一个支持客户/服务器模式的模型库管理系统的设计和实现,在这个系统中,模型被视为程序模块,通过运用面向对象的方法进行组织以及对模型的远程和动态的透明调用提供支持,使实现的系统具有良好的动态可扩充性.同时,通过提供模型定义语言和模型库管理,便于决策支持系统客户软件的开发.

关键词 决策支持系统,模型,模型库管理系统,动态调用,客户/服务器.

中图法分类号 TP391

决策支持系统DSS(decision support system)是在管理信息系统的基础上发展而来的,集成了模型库管理系统、数据库管理系统和会话部件,其主要任务是为高层管理的决策活动提供支持.决策支持系统通常解决的都是半结构化或非结构化问题,它的设计和运行是以模型驱动的.与通常的软件系统不同,决策支持系统的动态可扩充性非常重要,需要不断添加新的模型,因此,在决策支持系统中,我们必须提供一个框架,即模型库管理系统,供用户在运行过程中生成新模型、组合模型和动态运行模型.在数据库管理系统中,数据是被动的,而在决策支持系统中,模型具有主动和被动两种特性,模型不仅仅是存储问题,还涉及到如何定义其调用接口.如何支持模型的动态调用等问题.在Client/Server模式下还需考虑异种机、异种操作系统下模型的通讯、组合及运行控制,因而其管理也更加复杂.

模型管理技术经历了3个发展阶段[1]:程序文件系统、模型软件包和模型库管理系统.模型库管理系统建立了模型有效组织机构,可以对模型进行存储、修改、查询和调用,而且提供了模型组合功能.目前,国内开发的决策支持系统很少采用Client/Server,通常均是基于单机(包括多用户)系统的.而且模型的粒度往往也较大,是可执行文件,模型间的通讯往往是通过数据文件或公用通讯区的形式进行的,因而模型的运行效率较低,模型的组合也很困难.而一些基于静态函数库(Library)的系统往往可扩充性较差,难以在决策支持系统开发完毕后增加新的模型和知识.

本文介绍一个基于曙光一号并行机(UNIX操作系统)和PC机(MS Windows系统)的决策支持系统的模型库管理系统.该系统充分利用了曙光一号很强的事务处理能力和PC机Windows系统友好界面,能够较好地支持Client/Server模式的决策支持系统的开发和运行.该模型库系统支持两种粒度的模型(文件和函数子程序),包括完整的模型管理功能,同时还支持模型的动态调用和静态链接,使系统具有良好的可扩充性.

模型库管理系统面向客户服务器模式的应用,为本地和远程客户提供应用程序接口,其层次结构如图1所示.基于模型库管理系统,用户可以使用多种高级程序设计语言和系统,方便地开发具有良好开放性的(分布式)决策支持系统.

本文首先介绍模型定义语言MDL(model define language);然后以此为基础,详细讨论我们设计和开发的模型库管理系统,包括模型的组织和存储、模型的调用和组合等;介绍模型操纵语言MML(model manipulate language);最后给出总结和进一步的工作方向.

* 本文研究得到国家863高科技项目基金、国家自然科学青年基金、中国科学院软件研究所计算机科学开放实验室开放课题基金和香港城市大学资金资助.作者**李京**,1966年生,博士,副教授,主要研究领域为面向对象软件设计方法和环境,网络和分布式系统,多媒体系统.**孙颖博**,女,1965年生,讲师,主要研究领域为控制系统设计,应用软件开发.**刘智深**,1970年生,硕士,主要研究领域为网络,多媒体和软件工程环境.**张道一**,1937年生,教授,主要研究领域为分布式系统模型,验证和测试.

本文通讯联系人:李京,合肥 230026,中国科学技术大学

本文1997-04-23收到原稿,1997-07-18收到修改稿

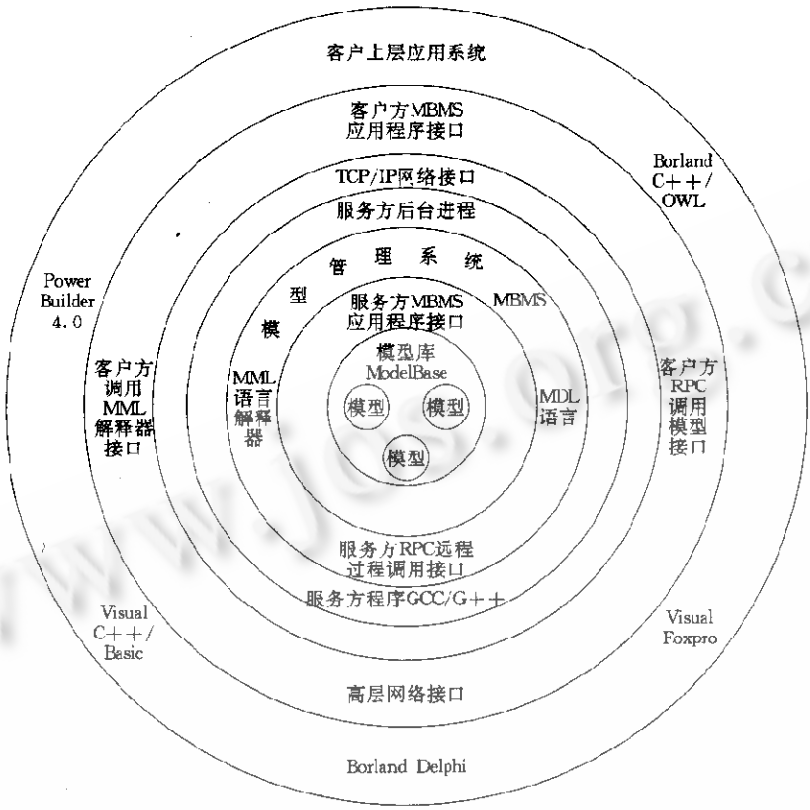

图1 C/S模式模型库管理系统构成及其接口示意图

1 模型表示与模型定义语言 MDL

模型是对现实的事物、现象、过程或系统的抽象描述，而方法是模型求解的算法.两者是多对多的关系.在我们的决策支持系统开发平台的设计和实现中，将模型与方法统一起来，直接用算法程序来表示模型，将模型库与方法库统一为模型库.

我们将模型定义为一个程序模块，模型是用 MDL 模型定义语言来定义的.MDL 遵循规范和实现相分离的原则，一个模型定义包括模型规范和模型实现两个部分.在模型规范中登记模型各种信息，包括模型名、所属模型库、模型描述、模型接口、模型关键字引用的子模型、模型动态链接所需要的目标文件与连接库文件信息等等内容.模型实现则是使用具体的程序语言(如 C,Fortran,Pascal,C++等)，按照一定的算法对模型规范的具体实现.模型规范可以实现为类、函数或子程序，也可实现为一个完整的程序.模型实现包括源程序、中间代码文件和生成的目标文件.

模型定义语言 MDL 的详细语法见文献[2]，它的简单形式如下：

```
ModelSpec 模型名 in 模型库名
    Description   模型语义的非形式化描述节
    keywords      关键字描述节
    SubModel      引用子模型列表节
    Interface     模型接口描述节
    Linkage       链接信息描述节
End
```

·Description 节可以给出模型语义的非形式化描述，如模型功能、用法、使用条件、版本、流程等，它有助于模型的理解，也可以用于模型的查询；

·keywords 节列出有关本模型的若干关键字信息，关键字可用于模型查询；

·SubModel 节用于定义复合模型情形，该节将本复合模型所引用子模型以及子模型所在的模型库信息一一对应

地进行说明,这些信息将为复合模型的动态调用提供支持.MDL 编译器将这些信息存入模型的子模型字典中,当模型要被系统动态调入时,相关子模型的所有链接节描述的目标文件、库将与本模型的目标文件、连接库一起被调入内存,以解决模型函数与服务方进程进行动态链接时的外部函数或全局变量的引用问题.

·Interface 模型接口节是模型定义最重要的部分,它用于描述模型所提供给外部服务的接口.模型服务接口分为3种,SHELL、函数和过程.前者实现为一可执行程序,然后通过 SHELL 接口调入,调用参数只能通过命令行参数传递,结果的返回不确定.函数和过程采用类 C 语言函数原型表示方法说明,除了给出参数的类型外,还需指明其输入输出特性(即 IN,OUT,INOUT).对于函数还要声明函数的返回值类型.函数和过程的语义则是通过使用一阶谓词逻辑给出函数或过程的前后条件来描述的.一个 Interface 节可以描述多个函数或 SHELL 接口.

·Linkage 节描述模型的目标文件及其引用的链接库信息列表及其它模型体目标文件,为模型的复合组装和运行提供支持.

下面以投资决策模型中净现值模型为例,给出用 MDL 定义的模型规范.净现值模型 NPV(net present value)调用了求和函数 Sum,该函数实现于平均值模型中(存放在 PrimitiveModel 模型库中),因而 NPV 模型是一个复合模型,模型的引用关系体现在模型 NPV 模型规范的子模型节中.以下忽略了 Interface 节中的语义描述.

```
Model "NPV" In "Investment" {
    DESCRIPTION {
      [净现值法:]
       (1) 说明:净现值是指投资方案未来现金净流量的现值与其原始投资现值之间的差额.
       (2) 公式:NPV=SUM(Rt/(1+i)EXP(t),n)-PC
           其中 SUM():求和函数,NPV:净现值(Net Present Value)
       (3) 接口:float ComputeNPV(rt0,n,i,p0,delta)  /*浮点净现值*/
    }   参数和返回值描述...
    KEYWORDS {"NPV";"净现值";"贴现率"}
    SUBMODEL {"Sum_&_Average" IN "PrimitiveModel";}
    INTERFACE {
      function {
        float ComputeNPV(rt0,n,ixl,p0,delta);
        float rt0[n];INOUT;long n;IN;float ixl;IN;float p0;IN;float delta[n];OUT;
      }
    }
    LINKAGE {
      COPYOBJ("./npv.o");
      LINKOBJ("/home/model/work/sum_average.o");
      LINKLIB("/usr/lib/libm.a")
    }
}
```

上面 MDL 文本定义模型 NPV 的规范,这里忽略了 NPV 的具体实现.NPV 模型存储于模型库 Investment(投资模型库)中.

在用户使用模型和组合模型时,通常仅需对模型规范进行理解,无需关心模型的具体实现.

2　模型库与模型存储结构

模型使用 MDL 定义,并使用具体的程序设计语言实现,通过调用 MDL 编译器将新的模型加入到指定模型库中,由于一个模型涉及众多的信息和文件,本节讨论如何管理、组织和存储模型.

模型表示为多个文件或字典形式,有模型的描述文件、模型接口文件、模型的关键信息字典以及模型动态链接信息字典、相关子模型信息字典.我们用 GNU 的 gdbm 来构造 MBMS 中的字典.

模型采用多重字典的方法,利用 UNIX 的文件系统管理,建立模型库的3层存储结构(如图2所示).顶层目录(设为/mbms)下存储整个系统的模型库字典,该字典中存放各个模型库的路径信息.第2层目录模型库存放模型库(设为mbl)模型字典,该模型字典中记录了该模型库中所有模型存储的路径,每个模型库存放于一指定路径下.第3层目录用来存放某一具体模型的所有相关信息,包括描述节文件、接口文件、关键字字典、子模型字典、中间目标文件与库文件字典等指针信息.

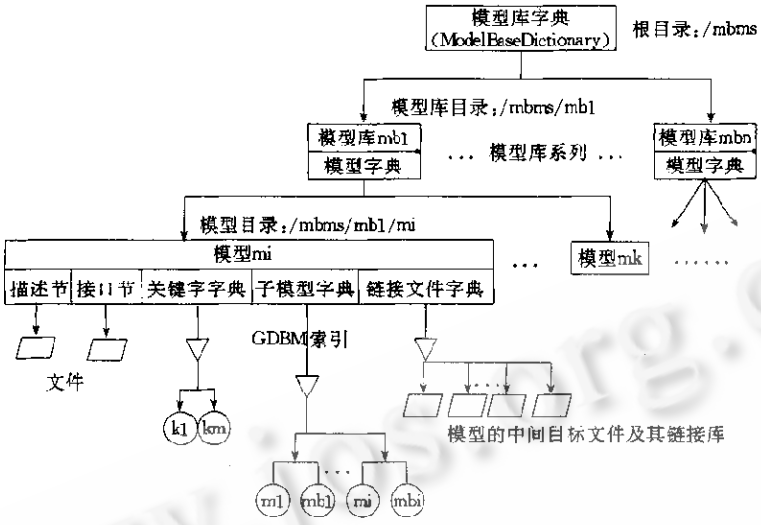

图2　模型库及其模型的字典组织结构

模型存储管理系统是按面向对象的方法实现的.从面向对象的观点来看,一个模型可以被视为一个对象,我们将模型属性与操作封装于一个模型类中.类中的属性包括模型的描述文件指针、关键字链表、子模型及其所在模型库列表、模型动态链接目标文件与链接库文件信息链表、模型接口函数或过程链表、接口函数的参数链表、数组参数的下标表达式链表等;类中的操作包括了模型的各项合法性检查(参数类型检查、数组参数下标表达式及其类型的合法性检查、子模型合法性检查、模型链接文件合法性检查)、模型STUB动态接口函数的自动生成和模型提交等.

3　模型链接和运行

由于我们要求模型库管理系统同时支持本地调用和远程调用,因而,模型的链接和运行也较为复杂.本节介绍将各独立模型的目标文件链接(静态或动态)成一个完整系统所涉及的主要技术.

3.1　模型的运行

模型有两种粒度,对于可执行文件形式的模型,其调用形式较为简单,使用函数Run来执行.下面,我们主要讨论函数形模型.对于函数形模型,必须经过链接才能运行.函数形模型提供两种接口,其中之一是普通C接口,它提供同一地址空间的函数模型的链接组合.另一种接口是用来进行不同地址空间(客户/服务器模式下)模型的链接或在动态调用模型时使用.

为了支持客户服务器模式,模型库管理系统启动一后台进程监控客户请求.一旦接收到请求,服务进程便进入服务状态,调用相应模型提供服务.具体过程如下:

(1) 服务进程接收到客户请求,获得被调用模型的有关信息,如模型库名、模型名、函数名及其输入参数等;

(2) 根据获取的参数类型及其值,构造被调用模型的参数链表(使用底层的类型流、数据流分解技术);

(3) 如果被调用模型已在内存,则执行第(4)步;否则,在相应模型库中查找所调用模型的外部目标文件,通过系统动态链接函数将其加载装入内存;

(4) 取得被调用模型的函数指针,将参数链表头指针传递给该函数,执行之;

(5) 将模型的执行结果返回客户.

由于服务进程无法预先得知所需调用的模型名及其参数的个数和类型,因而我们必须为所有的模型提供一个统一的调用接口.动态调用(Dynamic Invocation)模型与此类似,也需要一个固定的模型调用接口.这个模型的固定调用接口是系统根据模型规范自动生成的.假设被调用过程已经调入内存,且与服务方进程动态链接好后,从符号表中取得该被调用过程的函数指针,假设该指针命名为pRpcFunc,此时,只需要调用pFunproc(参数链表的头指针),也就是说,被调用过程的接口必须定义为固定的接口,其所需要的参数只能通过参数链表方式传入.我们根本无法用静态的程序描写出调用动态可变参数的过程,因此,只能使得被调用过程的接口相对于服务方进程是固定的,即只有固定的

入口参数链表头指针参数.而相对于过程本身,它的接口是不定的,因为它可以从服务进程传入的参数列表中获得可变参数,这与可变参数调用,在某种程度上是等效的.这样,在我们的服务方程序中静态地写出调用语句为 pFunproc(参数链表的头指针).

3.2 模型接口函数的自动生成

从上节讨论中看到,系统动态调用的模型接口是固定的,假如让程序员必须按照该接口函数,手工进行从参数链表到自己所需参数的转换,且手工将函数的返回结果反馈给客户方,是很繁琐的,也是不实际的.为使程序员在编写的调用过程摆脱复杂的参数传递、网络编程,而集中于过程本身的服务,我们为被调用过程自动生成一个接口程序,动态生成一段调用代码,该调用代码负责进行参数的传递,调用程序员所写的原过程以及结果的返回,以实现客户与服务方的远程过程的透明调用.程序员可以采用任何语言自由编写程序实现模型,生成目标代码,而 MDL 编译器在处理模型规范时,根据 Interface 节为该节描述的每个函数自动生成一个调用接口.下面我们给出接口的生成算法.

假设有模型函数或过程 funproc,则系统自动生成的调用接口函数的原形如下:

```
extern "C" {
  errcode funproc_stub(TypeDataStruc * pNetParamLinkHead);
}
struct TypeDataStruc {
  int       elementType;          /* 单元数据类型标识 */
  int       elementNum;           /* 单元数据个数,>1则为数组 */
  int       elementSize;          /* 单元数据字节数 */
  char      *pTypeData;           /* 数据起始地址 */
  struct    TypeDataStruc *next;  /* 指向下一类型数据 */
};
```

我们将该接口函数的形式实现为固定的,以便能够支持模型的动态调用.如果调用成功,则该 STUB 接口函数返回值为零,否则,为相应的出错代码.而参数仅仅有一个 pNetParamLinkHead,这是一个参数链表的头指针,该链表指向一个由服务进程从客户方接收到的实参数链表,该链表应与调用模型接口函数所有类型声明为 IN,INOUT 的参数及其数据类型一一对应.

接口函数自动生成算法.

假设为模型中的函数或过程 name 生成接口函数.

步骤1.　输出函数 name_stub 的原型声明,并用 extern "C" {...}抑制 C++的函数名变形;

步骤2.　生成函数头 int name_stub(struct TypeDataStruc *pNetParamLinkHead);

步骤3.　根据模型库中相应模型的规范,获得其模型函数的参数说明,然后输出参数动态类型检查和分割(Unmarshalling)程序;

步骤4.　调用模型接口函数,换名变量实参到函数形参的映射;

步骤5.　将函数的返回值和与模型接口中声明为 OUT 或 INOUT 的参数对应的换名变量通过 DSS 系统提供的网络 API 接口函数传回给客户方进程;

步骤6.　释放为数组变量分配的内存.

3.3 组合模型运行机制

模型可分为单元模型和组合模型.所谓单元模型是指不需调用其他模型的模型.组合模型则是通过本身模型调用其他单元模型或组合模型来构成.

我们开发的 DSS 系统提供组合模型的运行机制,作为解决复杂问题的手段.在 MDL 语言模型规范的子模型节中描述模型之间的连接关系,这样便可以定义一个模型网,该模型网即构成一个组合模型,模型之间的关系体现在模型程序之间的相互调用.由于程序语言具有丰富的控制结构,因此利用这种方法可以构成非常复杂的复合模型.它的缺点是模型组合需要编程.因为在开发的 Client/Server 模式的 DSS 系统中,模型是动态调入内存执行的,对于单元模型,由于它不与其他模型发生关系,动态调入内存很简单,只需要在模型链接文件节中,将需要系统调入内存的目标文件与链接库文件列出即可.但是,对于组合模型却不能简单地这样处理,因为模型之间存在调用,即外部引用关系.在系统提供的动态链接中,为了将某一目标文件动态调入内存进行链接,就必须首先解决模型之间的外部函数以及全局变量的引用,组合模型调用子模型,而子模型又可能调用其他子模型,并且模型调用关系可能形成"环".欲将组合模型动态调入内存,必须按照以下步骤进行.

将本模型引用所有的子模型,包括所有子模型所引用的其他子模型的所有需动态链接的文件,组装成目标文件和链接库文件两个链表.如果模型之间的引用形成"环",系统就必须提供机制识别模型环,以避免进入死循环;

服务方后台进程首先根据目标文件链表,将所有目标文件用 ld 命令预先链成一个临时目标文件,再将此临时文件以及所有的链接库链表作为参数传给系统动态调入函数 dynload,将其调入内存;

如果成功,则可从当前进程的符号表中取得组合模型的STUB接口函数的函数指针,然后将客户传来的实参链表传给该函数指针.执行该STUB接口函数,即可完成组合模型的动态调用运行.

4 模型操纵语言MML

模型库中存放着大量的模型,根据前面模型库的组织结构形式,为了查询模型,首先查询模型库字典,获得模型库的路径;然后查询模型字典库,获得需要的模型目录;最后,沿着该模型文件的存取路径查到相应的模型文件.模型库的查询过程包含两部分内容:一个是模型字典库的查询,由于模型字典库使用关系数据库建立,因此,其查询类似于关系数据库查询;但是由于模型文件并非存放在关系库中,因而最终模型的获得是通过操作系统的文件系统来完成的.

模型库还存在维护和管理问题,即模型的创建、删除、修改更新等功能.模型库的维护即是对模型库字典、模型字典的维护以及对相应路径的模型文件做适当操作.

我们在模型库管理系统中通过提供MML模型操纵语言来完成模型库的查询和维护.目前,我们仅实现了一个简单的MML,它只提供了对模型库和模型的最基本操作,还有待于扩充完善.用户使用MML解释器即时解释用户输入的MML命令,而MML命令可分为以下几类:(1) 模型库工作区的管理,工作区是用户操纵模型库的一个框架,用户在对模型库操作之前必须首先选择当前工作区.这部分命令包括工作区列表、别名设置和工作区选择.(2) 模型库的管理,包括模型库的创建、删除、打开、关闭以及模型库内容搜索等.(3) 模型管理,包括模型列表、查询、删除、显示模型描述、显示模型接口、显示模型链接文件链表、显示子模型链表、添加和更新模型等操作.(4) SHELL命令调用提供对操作系统命令的调用.(5) 系统命令是有关MML解释系统的命令,如帮助、退出系统等.

MML解释器支持客户服务器模式,既可以在本地终端上使用,也可以在客户方通过TCP/IP协议远程使用MML解释器,操作模型库与进行模型管理.

5 结束语

本文介绍了一个基于曙光一号并行机和PC486的客户/服务器模式的模型库管理系统,该系统使用C++,采用面向对象技术实现,具有方便用户开发应用系统、支持透明远程模型调用和动态模型执行等特点.通过使用这个开发平台,我们在863项目中已经成功地实现了一个开发区智能决策支持系统的原型.进一步的工作包括:(1) 扩充系统支持的模型表示形式,如模型的数据表示、逻辑表示等.(2) 扩充MML语言的功能,丰富开发工具,以更好地支持DSS整个开发周期.目前,我们仅提供了API库、开发语言的编译器和简单的模型库管理系统的交互环境.

参考文献

1 陈文伟.决策支持系统及其开发.北京:清华大学出版社,1994
(Chen Wen-wei. Decision Support System and Its Development. Beijing: Tsinghua University Press, 1994)
2 刘智深.客户服务器模式决策支持系统开发机制的研究[硕士论文].合肥:中国科技大学,1996
(Liu Zhi-shen. Research on the support of client/server decision support system development[M. S. Thesis]. Hefei: University of Science and Technology of China, 1996)

Design and Implementation of a Model Base Management System

LI Jing[1,2] SUN Ying-bo[1,2] LIU Zhi-shen[1,2] Cheung To-yat[3]

[1](*University of Science and Technology of China Hefei 230026*)
[2](*Laboratory of Computer Science Institute of Software The Chinese Academy of Sciences Beijing 100080*)
[3](*City University of Hong Kong Hong Kong*)

Abstract The MBMS(model base management system) is the kernel of a DSS(decision support system). The authors introduce the design and implementation of a client/server model base management system that runs on a heterogeneous distributed environment in this paper. In this system, a model is viewed as a program module and defined by a model IDL (interface definition language). By being organized with object oriented method and supporting remote and dynamic function (model) invocation transparently, the DSS may be developed easily and have very good static and dynamic extensibility. Also since many semantic information have been stored in the model base, the authors may develop more tools based on it to support the DSS development process.

Key words Decision support system, model, model base management system, dynamic invocation, client/server.