doi: 10.7690/bgzdh.2015.03.024

分布式通用模型管理

赵 华
（中国人民解放军 75753 部队，广州 510600）

摘要：为解决目前模型管理系统难以在网络环境下对多种文件格式的模型资源实施统一有效管理的问题，根据模型资源现有的各种技术体制，建立了一个通用模型管理系统的整体结构，并从模型描述与服务管理子系统、模型信息管理子系统及统一模型库和模型客户端介绍了系统的实现以及关键技术。结果表明：该系统能支持网络环境下多种文件格式模型的组合与服务，提高了多模型辅助决策的实效性。

关键词：决策支持系统；模型库；模型辅助决策
中图分类号：TP311.5 **文献标志码**：A

Management of Distributed General Model

Zhao Hua
(*No.* 75753 *Unit of PLA*, *Guangzhou* 510600, *China*)

Abstract: In order to solve the problem of the model management system which can't provide the effective management for the models in different formats in network at present, this paper design a general model management system and introduce how to realize the system and the key technology from model description and service management subsystem, model information management subsystem and unified model base and model client. Interpretation of results, the system provides general management and service for models in different formats in network, and advance availability of the multi-models-aided decision.

Keywords: decision support system; model base; model-aided decision

0 引言

模型库管理系统作为决策支持系统(decision support system，DSS)的三大部件之一，得到了广泛关注和研究[1-6]，研究人员基于多种模型表示方法、结合先进的计算机技术，提出了各种模型管理与服务技术，在基于模型的结构化决策问题求解方面取得了一系列成果，但目前难以在网络环境下针对多种文件格式的模型资源实施统一的管理，而且缺乏一套行之有效的解决方案，降低了系统的可操作性。

针对这些不足，笔者设计实现了一个模型管理与服务系统，针对决策问题，提供自顶向下和自底向上 2 种解决方案，同时支持网络环境下标准动态链接库、COM 组件、标准应用程序和 Web Service 等模型资源的统一管理，为模型库管理系统提供了一种新的设计方案。

1 总体设计

为了充分利用网络资源，在逻辑上建立庞大的模型库，同时保证对这些模型的合理使用，系统需要针对模型提供者和模型使用者的不同需求，提供相应的模型注册和模型使用服务。系统的整体结构如图 1 所示。

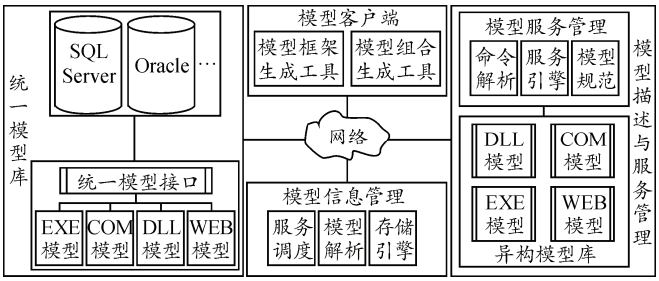

图 1 体系结构

1) 模型描述与服务管理子系统：面向模型提供者，提供模型注册功能。支持以标准动态链接库、COM 组件、标准 EXE 以及 Web 服务为载体的模型文件。通过解析用户所制定的模型，提取出对应的模型方法信息，按照预定的统一模型规范，针对不同类型的模型，由模型规范模块将其转化为统一模型文件，并根据用户提供的说明信息、提供端信息及模型方法信息生成描述文件，作为外界调用规范，是模型的统一接口。服务引擎支持外界调用模型的实际运行，通过调用模型的统一接口，接收调用信

收稿日期：2014-09-23；修回日期：2014-11-20
作者简介：赵 华(1984—)，男，浙江人，硕士，从事系统规划与管理决策技术研究。

息，实现命令解析，调度相应模型的统一模型文件，实现运行，并将结果通过统一接口反馈。模型描述与服务管理子系统是实现异构模型向统一模型转化的核心部件，模型的统一接口是外界通信的规范。

2) 模型信息管理子系统：负责管理模型信息，更新注册模型的统一接口，并记录模型的调用情况。通过接收模型描述与服务子系统发送的注册模型接口文件，进行试运行，并根据已有的模型信息，创建模型的 ID 编码等模型方法的基本信息，同时更新数据库，构建统一模型库。模型信息管理子系统支持多个模型描述与服务子系统的注册信息，需要协调注册顺序、避免重复注册和处理信息更新，提供模型分类和模型查询等功能，辅助模型信息管理。

3) 统一模型库：利用成熟的数据库技术实现模型库的构建。统一模型库中的模型并非实际的模型文件，而是接口信息，本身没有计算能力，是调用模型的逻辑体现。由模型信息管理子系统解析模型接口文件，将模型方法的各种信息按照预定的编码规则存入数据库，完成统一模型库的构建。任何模型的调用需要根据统一模型库中的接口信息进行调用报文的创建，实现与模型描述与服务子系统的通信。统一模型库允许多个模型注册和调用，需要处理脏数据，保证注册与调用的一致性。

4) 模型客户端：面向模型使用者，提供 2 种解决方案，实现多模型的辅助决策。通过模型组合生成工具或模型框架生成工具，按照自底向上或自顶向下的思路构建解决决策问题的组合模型结构，利用统一模型库中的信息，创建调用报文，实现模型调用。模型客户端使用的模型必须保证与统一模型库信息的一致性，同时需要对于各个模型的执行流程及参数传递进行控制。

系统围绕模型注册和模型调用 2 个核心功能展开，通过模型描述与服务管理子系统实现异构模型的规范化，利用模型接口文件，由模型信息管理子系统实现模型的注册。通过模型客户端实现模型的组合，生成执行流程，利用统一模型库中的模型接口向对应的模型提供端发送调用命令，实现模型调用。所有模型接口信息作为模型的签名，存储在统一模型库中，是系统的信息中枢，实现了异构模型的同构化。

2 系统实现及关键技术

2.1 模型描述与服务管理子系统

模型描述与服务管理子系统辅助用户完成模型信息的描述，实现模型的注册，同时负责接收模型服务请求并调度模型运行。作为模型提供端使用的工具，应当具备以下功能：1) 多种载体格式模型的原始信息提取与模型规范化信息描述；2) 监听网络模型使用方的连接请求，建立与模型使用方的网络连接；3) 调度多种载体格式模型运行，负责记录、管理模型运行结果和日志；4) 根据输出参数传递方式将模型运行结果返回模型使用方。

为了实现多种载体格式的模型注册必须通过模型规范模块进行模型的解析，目前系统支持以下 4 种类型的模型：1) 标准动态链接库模型文件，包括 Win32 和.NET Framework 下的动态链接库文件；2) 基于 COM 的模型文件，可以为 DLL 或 EXE 格式的组件文件；3) 标准应用程序类模型，即标准 EXE 文件；4) 基于 Web 服务的模型。实现的文件格式不限，通过 Web 服务提供。

上述模型载体形式均是 Windows 操作系统支持的文件格式，对于其他操作系统中的模型载体形式暂不考虑。

对于上述 4 种主要的模型载体形式，在模型描述和注册中，为了减轻工作量、提高效率，同时确保模型信息的准确性和一致性，系统提供自动或半自动的手段，尽可能地从模型资源文件中直接提取原始的模型方法和参数等关键信息，其中模型方法信息主要有模型名称、功能描述、研发单位和输出参数类型等；模型参数信息包括参数类型、名称和含义等。

模型信息的提取主要指从模型文件的元数据中提取函数的名称、输出参数类型和输入参数的信息(包括类型、名称以及顺序)等信息。针对不同文件实体的模型，提取方法也各不相同，但主要是利用 Visual Studio 2005 开发平台中的反射技术(Reflection 命名空间)和一些工具程序(例如 TlbImp.exe 和 dumpbin.exe)。

2.1.1 关键技术

反射(Reflection)是.NET 中的重要机制，通过反射，可以在运行时获得.NET 中每一个类型(包括类、结构、委托、接口和枚举等)的成员，包括方法、属性、事件，以及构造函数等，同时根据这些元数据实现实例的创建和调用。因此，通过使用反射技术，获取注册模型封装的程序集，从而实现模型信息的提取功能，同时还可以动态地创建模型实例，调用相应的模型方法。

Tlbimp.exe 是一个.NET Framework SDK 包含的名为类型库导入程序的工具，该工具通过围绕组件，生成托管包装器，将 DLL 文件中的标准 COM 组件转换为等效的.NET Framework 程序集。转换后的组件可以早期绑定到托管代码，这样将大大提高性能。

Dumpbin.exe 是 Microsoft COFF 二进制文件转储器，显示有关 32 位通用对象文件格式(COFF)二进制文件的信息。可以使用 Dumpbin 检查 COFF 对象文件、标准 COFF 对象库、可执行文件和动态链接库(DLL)中的文件及其函数信息。

wsdl.exe 是一个.NET Framework SDK 包含的 Web 服务实用工具，使用 ASP.NET，根据 WSDL 协定文件、XSD 架构和.discomap 发现文档，为 Xml Web Services 客户端和 Xml Web Services 生成代码。

2.1.2　模型的信息提取和规范化

模型信息的提取是为了方便用户注册，主要是针对模型方法的函数信息，包括对动态链接库、COM 组件和 Web 服务 3 类载体形式的模型。

1) 动态链接库模型。

动态链接库模型分为 Win32 DLL 和.NET Framework DLL 2 种类型，其中 NET Framework DLL 可以直接使用反射技术得到其函数信息，而对于 Win32 DLL 需要借助 Dumpbin.exe 得到函数信息的 TXT 文档，针对该文档使用 StreamReader 的相关函数提取出模型信息。

规范 Win32 DLL 需要根据提取出的模型信息，生成代理 cs 文件，编译为.NET Framework DLL 托管文件，再使用反射实现模型方法调用。例如对于 Win32 DLL 模型文件 example.dll，内有函数：int add(int a, int b)，系统需要生成如下 cs 文件：

```
//系统自动生成以下代码，存储在 temp.cs 文件中
using System;
using System.IO;
using System.Runtime.InteropServices;
using System.Collections;
using System.Windows.Forms;
public class Win32Dll
{
    [DllImport("\example.dll",EntryPoint="add",
    SetLastError=true,CharSet=CharSet.Unicode,
    ExactSpelling=true,CallingConvention=CallingC
    onvention.StdCall)]
    public static extern int add(int a ,int b);
}
```

动态编译此文件生成托管的.NET Framework DLL 文件。再利用反射技术实现调用。

2) COM 组件模型。

注册 COM 组件模型时，首先利用 Dumpbin.exe 工具生成包装器类，该包装器类是 1 个.NET Framework DLL 托管文件，然后利用反射实现信息提取和模型调用。

3) Web 服务模型。

Web 服务模型可以通过服务描述文件(.wsdl)注册，也可以提供服务地址实现注册，它的信息提取及调用是通过 wsdl.exe 工具生成 CS 代码文件，经过动态编译生成以.NET Framework DLL 为载体的 Web 服务的代理类，最后使用反射技术实现。

模型的规范化是为了方便异构模型服务的调用，将各种不同类型的模型转化为.NET Framework 的动态链接库 DLL 文件，再使用反射实现调用，具体过程如下：

```
//使用反射调用.NET Framework DLL 模型方法
    assembly = Assembly.LoadFile(托管文件路径
名);
//加载指定路径上的程序集文件的内容。
    Type type = assembly.GetType(类名);
//获取表示指定类型的 Type 对象
    MethodInfo method = type.GetMethod(函数
名);
//获取当前 Type 的特定方法。
    Object obj = Activator.CreateInstance(type);
//创建指定类型的实例。
    result = method.Invoke(obj, 函数参数信息);
//调用由此 MethodInfo 实例反射的方法或构造
函数。
```

2.2　模型信息管理子系统及统一模型库

模型信息管理子系统与统一模型库针对注册模型的接口描述文件实施管理。统一模型库中的模型接口描述是系统中的信息中枢，参与模型注册、模型调用的整个过程。它的逻辑结构为：

模型类别(类别编码，类别名)；

模型(模型 ID，类别编码，模型文件类型，模型文件路径，模型提供端地址，模型描述)；

模型方法(模型方法 ID，模型 ID，模型方法描述，类型名，函数名)；

参数(模型方法 ID，参数序号，参数类型，参

数名，参数描述)；

模型信息管理子系统负责模型描述信息的注册、注销、查询、更新以及模型的测试运行等功能。

1) 模型描述信息的注册：根据模型描述规范，将规范化的模型接口信息注册到统一模型库中；

2) 模型描述信息的注销：根据模型管理或模型服务提供方的共享服务需要，注销统一模型库中的模型接口信息。

3) 模型描述信息的查询：根据模型用户的查询请求，对统一模型库中的模型接口信息进行查询和筛选。

4) 模型描述信息的更新：根据模型服务端的要求，对统一模型库中的模型接口信息进行更新和升级。

5) 模型测试运行：根据统一模型库中的模型接口信息，负责提交模型的测试运行请求，接收并展现运行结果。

6) 模型分类管理：支持用户根据需求动态建立、修改模型分类体系，便于模型的分类管理和应用。

2.3　模型客户端

从 2 个方面支持模型重用：1) 以功能聚合为导向，提供以搭积木的方式自底向上聚合现有模型的功能；2) 以决策问题为导向，提供自顶向下层层分解决策问题、最终生成问题求解框架的功能。模型客户端使用的脚本利用 EBNF 编码格式的 XML 文档描述，并在控制脚本生成前对组合模型的可操作性和参数完整性进行检查。

2.3.1　模型框架生成工具

模型框架生成工具由问题分解模块和脚本生成工具组成。

1) 问题分解模块。

问题分解模块为用户提供分解问题的可视化工作界面，以决策问题为核心，逐层分解，针对能够解决的子问题进行模型匹配，直到形成最终的组合模型。在分解过程中，问题之间通过顺序、选择和循环 3 种控制关系相互链接，并通过信息流实现输入输出参数的类型转换。向用户提供的图元包括开始、结束、决策问题和 3 种控制关系，其中，每个用户视图具有唯一的 ID 号，一个用户视图中的开始和结束匹配产生并且唯一，循环和选择的开始与结束匹配产生，决策问题图元如果被分解，则记录其对应的用户视图 ID，否则进行模型匹配，同时对任何图元提供描述功能。

为了描述问题的层次关系，制定以下用户视图 ID 编码方式：层号_用户视图号。其中层号表示问题视图处于的层次，用户视图号是指在当前层上，该用户视图的序号。决策问题 ID 编码方式：所在用户视图 ID_问题序号。例如：决策问题分解的第一个用户视图编号为 1_1，其中有 2 个决策问题，ID 分别为：1_1_1 和 1_1_2，由 1_1_1 分解出的用户视图 ID 为 2_1，其中的问题 ID 依次为 2_1_1，2_1_2 等等，由 1_1_2 分解出的用户视图 ID 为 2_2，其中的问题 ID 依次为 2_2_1，2_2_2 等等。

2) 脚本生成模块。

根据用户视图，生成 XML 格式的组合模型控制脚本，并用一组 XML Schema 描述组合模型控制流和参数传递流，算法思想如下：

(1) 获取未生成控制脚本的用户视图，取得开始图元，创建以其 ID 号命名的 XML 文档并生成<CompositionProblem>，增加 ID 属性，表示用户视图的 ID，取下一个图元为当前图元；若所有用户视图均生成了控制脚本，则转(4)；

(2) 生成<Input>、<Output>和<Var>，分别描述输入参数、输出参数和局部变量的信息；

(3) 判断当前图元，如果为结束，则返回(1)；否则按以下匹配情况处理：

① 若为决策问题，设置与其相连的下一个图元为当前图元，创建<Problem>，如果被分解则生成属性 Sub，记录对应的用户视图 ID，否则记录其匹配模型信息包括其参数传递情况，转(3)；

② 若为选择，生成<IF>，并对其每个分支进行以下操作：生成<IF_Fork>，记录每个分支的条件，并设置与其相连的下一个图元为当前图元，转(3)；

③ 若为选择结束，生成<ENDIF>，设置与其相连的下一个图元为当前图元，转(3)；

④ 若为循环，生成<For>，并设置其循环条件，设置与其相连的下一个图元为当前图元，转(3)；

⑤ 若为循环结束，生成<ENDFor>，设置与其相连的下一个图元为当前图元，转(3)；

⑥ 若为顺序，设置与其相连的下一个图元为当前图元，转(3)；

(4) 设置当前脚本为 1_1，获取 1_1.xml 内容；

(5) 获取当前脚本内容，获取具有属性 Sub 属性的<Problem>标签，如果没有，则算法结束。否则，打开以 Sub 属性命名的 XML 文档，用其内容替换<Problem>，转(5)；

（下转第 96 页）