软构件技术在作物管理智能决策系统设计中的应用

朱 艳, 曹卫星, 王绍华, 潘 洁
(南京农业大学)

摘 要: 基于知识模型和生长模型的作物管理智能决策系统是一个综合了多领域的作物信息和专家知识, 集成了多方面相关支持技术的决策系统。因此, 如何设计好系统中的各功能模块, 使其便于非系统开发者对其精华部分的学习和理解, 同时又能即插即用, 利于系统的维护, 是系统成功与否的关键。以小麦作物为例, 论述基于软构件技术的作物管理智能决策系统设计中软件平台的选用、系统的构件化结构设计与实现过程以及数组在接口上的传递等关键技术, 从而为农业管理决策系统的开发提供一种构件化的基础框架。

关键词: 软构件技术; COM 接口; VC++ 软件平台; 作物管理; 智能决策系统
中图分类号: S126 **文献标识码**: A **文章编号**: 1002-6819(2003)01-0132-05

1 引 言

农业生产是一个涉及多学科、多技术交叉的复杂领域, 一个好的作物管理智能决策系统往往需要综合包括气象、土壤、育种、农学、计算机、管理等学科在内的知识和技术, 系统中许多有关作物生理生态的机理模型和生产管理的专家经验模型需要在系统应用过程中不断得到改进和完善。因此, 当一个真正的决策系统最终以软件的形式服务于用户的时候, 系统应该能在旧版本的基础上不断得到升级, 并且可以根据用户的需求方便地集成到其它更大的系统中去, 从而发挥其应有的作用。目前, 基于软构件技术的应用系统开发是软件业界流行的软件开发模式[1], 应用于作物管理智能决策系统还鲜见报道。本文以小麦作物为研究对象, 利用软构件的语言无关性、可重用性、简便快捷的系统维护机制等特点, 在 VC++ 平台上构建一个基于软构件技术的作物管理智能决策系统, 从而较好地满足系统开发的任务需求。

2 软构件的特点

目前, 软构件以其语言和操作系统的无关性、跨平台的软件复用和简便快捷的系统维护机制等特点而越来越得到软件开发者的青睐, 是将来软件开发的主流模式[1,2]。

1) 语言和操作系统无关性。软构件的开发遵循 COM/DCOM (Component Object Model/Distributed Component Object Model) 下的 Iunkown 和 IclassFactory 接口标准。COM/DCOM 为组件软件和应用程序之间进行通信提供了统一的标准, 也为组件程序提供了一个面向对象的活动环境。COM 标准中的规范不依赖于任何特定的语言和操作系统, 只要按照该规范, 任何语言都可使用。

2) 跨平台的软件复用。软构件以运行级二进制代码的形式提供, 并通过标准描述语言 CDL (Component Description Language) 向外界提供软构件的外部结构和内部行为信息。因此, 组件之间可以在二进制级别上进行集成和重用。

3) 便用性。在 COM 模型中, 客户请求服务通过接口进行。每一个接口都有一个 128 位的全局唯一标识符 (GUID, Globally Unique Identifier) 来标识, 客户通过 GUID 获得接口的指针, 再通过接口指针, 客户就可以调用其相应的接口函数。至于具体功能如何实现, 则完全由接口内部实现, 客户不必了解。

4) 简便快捷的系统维护机制。组件技术将一个庞大而复杂的应用程序分解成一个个组件模块。因此, 在进行程序修改或版本升级时, 就可以只修改或替换相关的组件, 而不影响其它众多的程序组件。

3 作物管理智能决策系统的工作流程

基于知识模型和生长模型的作物管理智能决策系统是由本实验室研制开发的一种新型的决策系统, 综合了生长模型的预测功能和知识模型的决策功能[3,4]。其中, 生长模型是在大量的试验和文献资料的基础上, 通过解析天气- 土壤- 技术措施- 作物生长过程的机理关系, 发展和建立起来的基于生理生态过程的作物生长系统模型[4]。知识模型是在收集和分析有关作物与环境关系及不同区域的模式化栽培技术体系等知识的基础上, 建立起来的描述作物生育指标地域性和季节性变化规律的量化关系[3]。基于知识模型和生长模型的作物管理智能决策系统以作物生长机理模型为作物生长动态及产量和品质形成的预测器, 以具有时空特征的作物栽培知识模型为决策的智能支撑, 以地区性土壤、气候、生产条件、作物品种等数据库为信息依托, 实现了播前方案设计和产中动态调控两大决策功能[4]。

3.1 播前方案设计

系统根据决策点的气候和土壤状况、用户的产量和

收稿日期: 2001-09-11
基金项目: 国家自然科学基金重点项目(30030090); 国家 863 计划项目(2001AA110520) 资助课题
作者简介: 朱 艳(1976-), 女, 博士生, 主要从事作物模拟与信息技术方面的研究。南京农业大学作物生长调控重点开放实验室, 210095。
Email: yanzhunau@hotmail.com

品质目标等, 通过匹配知识模型和生长模型, 制定出一套播前栽培方案, 包括品种、适宜的产量结构和品质指标、播期、基本苗、播种量、调控指标、肥料运筹及水分管理等(图 1)。

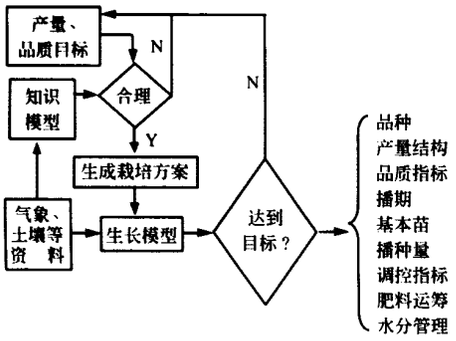

图 1　播前方案设计流程图

Fig. 1　Flow chart of pre-production plan

3.2　动态调控

系统以栽培方案中作物调控指标的适宜范围为标准“专家曲线”, 当生长模型预测的作物生长发育状况偏离曲线或与专家知识不符时, 系统分析原因, 推荐一个适宜的调控措施(如播种期、播种量、肥料与水分运筹等技术措施)以及调控时期。用户也可以在特定时期输入实际作物生育状况(可以是 RS 技术获得的苗情信息或田间考苗情况), 以提高下一阶段的预测性。系统最后输出决策的技术措施及预测的作物生育动态及产量和品质的形成[4](图 2)。

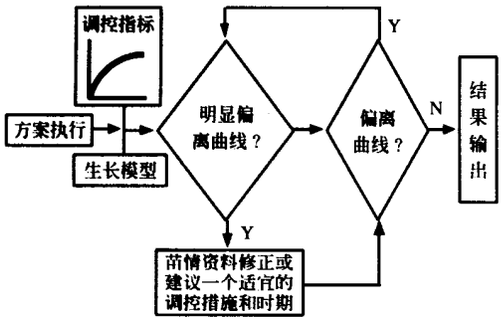

图 2　实时管理调控流程图

Fig. 2　Flow chart of real-time control and regulation

4　软件平台的选用

作物管理智能决策系统综合了包括气象、土壤、育种、农学、计算机、管理等学科在内的知识和技术, 其内在的逻辑关系相当复杂, 一般只被系统开发者所真正理解和明白。而且, 开发一个如此巨大的系统往往需要多个学科的研究人员(包括研究生、博士后等流动人才)合作几年才能完成。因此, 如果用传统的基于过程的编程语言来设计该系统, 则存在以下几个缺点[2, 5, 6]: 1) 长而复杂的程序不便于非系统开发者对其的理解和学习。2) 重复代码多。3) 系统中的数据和功能很难被其他系统所使用。4) 系统维护极不方便, 系统可能会随着开发者的退休而退休。

为了克服以上缺点, 我们采用了 VC++ 编程语言。VC++ 是 Microsoft 公司开发的基于 C/C++ 的面向对象语言, 它具有如下特点: 1) 面向对象的特征(类和对象、使用重载函数和重载符及多态性、继承性和良好的数据封装性等)使得系统能根据其内在的特点和功能进行模块化设计, 非系统开发者可以按类分模块对系统进行学习和理解甚至修改和完善, 其效率不言而预。2) VC++ 具有执行速度快、占用内存少等特点。3) VC++ 从 4.0 版本就已经提供了全面的 COM 支持, 尤其在 5.0 和 6.0 版本中, 不仅 MFC 类库提供了 COM 应用支持, 而且 VC++ 的集成开发环境 Visual Studio 也为 COM 应用提供了各种向导支持。对于各种建立在 COM 基础上的应用技术, 包括自动化对象、ActiveX 控件、OLE 服务程序或 OLE 包容器程序, Visual Studio 都提供了快速的向导支持[6, 7]。因此, VC++ 软件平台无疑是本系统设计的合适之选。

5　系统构件化结构设计

根据本系统的特点和功能, 我们将整个系统做成 3 个自动化软构件: 生长模型、知识模型和管理决策软构件。其中, 生长模型软构件包括生育期、干物质积累、形态发生与器官建成、干物质分配与器官生长、养分和水分平衡等六个接口函数; 知识模型软构件包括产量目标与结构、品质目标与指标、品种选择、播期确定、密度设计、肥料运筹、水分管理、理想生育期、主要生育指标动态、源库指标、植株营养诊断指标、土壤养分和水分动态等 13 个接口函数; 管理决策软构件包括方案设计和动态调控两大接口。同时, 部分接口函数又包含了若干个内部函数, 见图 3。在软构件中, 每一接口函数均包括数据的输入、输出和模型计算三部分。其中, 数据的输入、输出均放在接口上通过变体传递, 模型的具体计算在接口函数内部进行。因此, 非系统开发者可以按组件依次对系统进行理解和学习, 而对于那些不想了解系统内部关系、只想使用系统某一组件的客户来说, 只要按照系统开发者提供的接口函数输入输出说明, 就可直接调用接口函数。

6　自动化软构件在 VC++ 平台上的实现过程

由于组件外对象(.exe)自身调试方便, 并且很容易将其转化成 dll 程序。所以本文以 exe 程序为例, 简要介绍自动化软构件在 VC++ 软件平台上的实现过程, 包括自动化工程的创建、自动化类、接口函数和属性的添加及程序调试等。

6.1　自动化工程的创建

6.1.1　直接创建自动化工程

具体步骤如下: 1) File→New < 工程名 DSWM (EXE) > →点击 OK。2) 连续两次单击 Next。3) 其它支持< 选择 Automation 及 ActiveX controls> 。4) 单击 Finish 即可。

6.1.2　将普通工程转化为自动化工程

在工程的 stdafx.h 文件开始部分添加# include “afxole.h”, 并在工程的 cpp 文件中的 InitInstance() 函

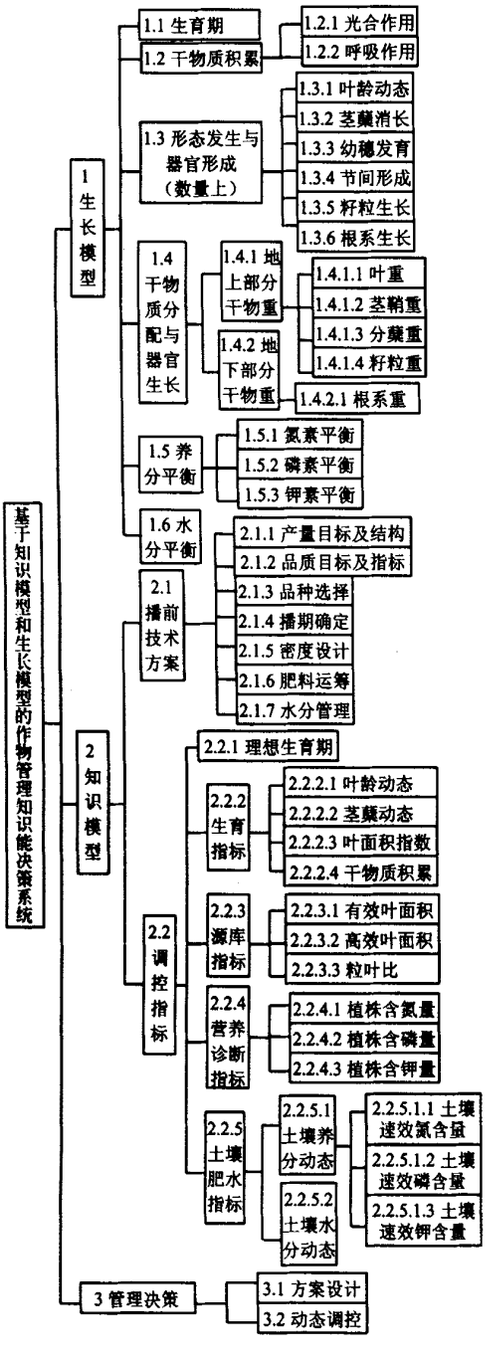

图 3 系统构件化结构图

Fig. 3 Component structure of system

数中添加函数 AfxOleInit() 即可。

6.2 为自动化工程增加自动化类

自动化类的添加步骤如下: 1) View→ClassWizard→Add Class→New。2) 在 Class Name 中填入类名, 在 Base Class 中选择一合适基类, 在 Automation 选项中选择< Created by type ID: > 并填入 ID 标识符(一般为"工程名. 类名")。3) 连续两次单击 OK 即可。

6.3 为自动化类添加方法(接口函数)

接口函数的添加步骤如下: 1) View→ClassWizard→Automation→选择需添加函数的工程及类? Add Method。2) 在出现的对话框中添加 External name(接口对外所提供的函数名)、Internal name(工程内部所使用的函数名)、Return type(函数返回类型), Parameter list(函数参数列表)。3) 连续两次单击 OK。

6.4 为自动化类添加属性(成员变量)

成员变量的添加步骤如下: 1) View→ClassWizard→Automation→选择需添加属性的工程及类名→Add Property。2) 在出现的对话框中添加 External name(接口对外所提供的属性名)、Variable name(工程内部所使用的属性名)、Type(属性类型)。3) 连续两次单击 OK 即可。

6.5 程序调试

在程序调试初期, 利用自身调试的可跟踪性, 对程序内部的语法、逻辑关系等细节进行调试, 在确认无问题后, 再用外部引用调试来测试各接口有无问题。

6.5.1 内部自身调试

内部自身调试即在同一工程内做一个调用函数, 调用时先定义一个指向接口函数所在类的指针, 通过该指针去调用接口函数, 调用完成以后用 delete 释放指针。

6.5.2 外部引用调试

此处主要讲述 VC++ 应用程序调用外部组件的具体过程: 1) 创建自动化工程。2) View→ClassWizard→Add Class→选择 From a type library→找到要引用的文件(EXE 程序为 tlb 文件, DLL 程序为 tlb 文件或 dll 文件)→Open→选择你所需的接口类→连续两次单击 OK 即完成引入。3) 实际调用接口函数时, 需首先调用创建接口函数 CreateDispatch("ID"), ID 名是创建接口类时填写的; 然后才能调用接口函数, 并且注意在调用结束后通过 ReleaseDispatch() 函数释放接口。

7 数组在接口上的传递等关键技术

7.1 数组在接口上的传递

在 VC++ 中, 接口不支持数组的直接传递, 而只能采用变体的方式来进行。本文主要讲述一维和二维数组在接口上的传递技术。

7.1.1 一维数组

1) 数组的传入

外部数组必须打包后才能传入到接口函数内部。假设现有一个 double 型数组 D[5], 经打包后传递到接口函数 BOOL inputfun(VARIANT FAR * array) 中, 其具体过程如下:

```
SAFEARRAY *m_pSafeA;                    //定义一个安全数组
SAFEARRAYBOUND rgsab[1];
                                        //定义数组的维数为 1
rgsab[0].lLbound= 0;                    //定义数组下标为 0
rgsab[0].cElements= 5;                  //定义数组中元素个数为 5
m_pSafeA = SafeArrayCreate(VT_VARIANT, 1, rgsab);
                                        //按照定义创建一个空的安全数组
VARIANT var;                            //定义一个变体变量
for(long i= 0; i< 5; i++)
//以下是将 D 中的元素打包到安全数组 m_pSafeA 中
{
    var.vt= VT_R8
    var.dblVal= D[I];
```

```
    SafeArrayPutElement(m_pSafeA, &i, &var);
}
VARIANT tarray;                    //定义一个变体变量
tarray.parray= m_pSafeA;           //将安全数组附给变体
model-> inputfun(&tarray);
        //直接调用接口函数,model 为指向接口函数
        所在类的指针
```

2) 接口函数内部要使用通过变体传入的元素, 必须将变体解包才行。其具体过程如下:

```
VARIANT vinput[5];                 //定义一个变体数组
long plUb, plLb;
for(long i= 0; i< 5; i++)
{
    VariantInit(&vinput[i]);
}                                  //将变体数组初始化
SafeArrayGetUBound(array-> parra, 1, &plUb);
SafeArrayGetLBound(array-> parray, 1, &plLb);
                                   //获取数组上标和下标
Long m= plUb- plLb+ 1;             //计算数组中元素的个数
for(long i= 0; i< m; i++)
{
    SafeArrayGetElement(array-> parray, &i,
&vinput[i]);
        //将变体 array 中的元素取出放到 vinput[]中
  D[i]= vinput[i].dblVal;
  //将变体数组 vinput[]中的元素值取出放入数组 D[]
}
```

3) 外部调用接口函数时, 如何取出接口上带出的变体中的数组元素　在调用接口函数以前, 先定义一个空变体, 并且给此空变体分配空间, 然后直接调用接口函数即可。其具体过程如下:

```
VARIANT array;                     //定义一个变体变量
array.vt= VT_ARRAY;                //给变体 array 分配空间
array.parray= m_pSafeA;            //m_pSafeA 为空安全数组
Model-> outputfun(&array);         //直接调用接口函数
    ……                             //将 array 解包后取出里面的元素使用
```

7.1.2　二维数组的传递

二维数组在接口上的传递与一维数组类似, 现就二维数组的打包过程进行举例说明。

```
SAFEARRAY *m_pSafeA;               //定义一个安全数组
SAFEARRAYBOUND rgsab[2];
                                   //定义数组的维数为 2
VARIANT var[3][5];                 //定义一个二维的变体数组
rgsab[0].lLbound= 0;               //定义数组第一维下标
rgsab[0].cElements= 3;             //定义数组第一维元素个数
rgsab[1].lLbound= 0;               //定义数组第二维下标
rgsab[1].cElements= 5;             //定义数组第二维元素个数
m_pSafeA = SafeArrayCreate(VT_VARIANT, 2,
rgsab);                            //按照定义创建一个空的安全数组
long index[2];
for(long j= 0; j< 3; j++)
{
  index[0]= j;
  for(long i= 0; i< 5; i++)
  {
    var[j][i].vt= VT_I4;
    var[j][i].lVal= (i+ 10)* j;
    index[1]= i;
    SafeArrayPutElement(m_pSafeA, index,
&var[j][i]);
  }
}
VARIANT fff;                       //定义一个变体变量
fff.parray= m_pSafeA;              //将安全数组附给变体
```

7.2　其它关键技术

7.2.1　Cstring 字符串在接口上的传递

Cstring 字符串必须通过函数"Cstring 字符串名.AllocSysString()"将其转化成BSTR 型才能在接口上传递。举例如下:

```
CString cd= "d:\data\NjCD.mdb";
BSTR cdming= cd.AllocSysString();
model-> fun(&cdming);              //调用接口函数
```

7.2.2　组件外对象(.exe)向组件内对象(.dll)的转化

具体步骤如下: 1) 创建一个支持自动化接口的 dll 工程, 把 exe 工程中的所有非 exe 自身的类全部添加到 dll 工程中。2) 修改 dll 工程中的.odl 文件, 保留对 dll 工程自身的说明, 把 exe 工程中的.odl 文件中对接口类的所有说明拷贝到 dll 工程中的.odl 文件中(在 dll 工程自身说明的下面)即可。

7.2.3　用VB 调用

VC++ 接口对象时, 二维以上数组应使数组维数前后对换, 以使数据传输正确。

7.2.4　接口设计过程中属性的使用

接口设计过程中, 我们发现属性的使用有些不太方便的地方。比如在系统维护过程中, 如果你想在接口类中增加(删除)一个属性, 这就意味着接口发生了改变, 客户必须重新引用才能使用。但是, 如果在设计过程中将要添加的属性统统放到变体中, 那么, 系统维护者只要将变体数组的上界加一(减一)就相当于增加(删除)了一个属性, 而此时接口并没有改变。因此, 如果可能的话, 尽量不要使用属性, 而统统用变体代替。

8　结　语

基于软构件技术的应用系统是软件系统产业化开发的理想模式, 是解决复杂软件应用系统开发的科学方法。在VC++ 开发平台上建立起来的基于软构件技术的作物管理智能决策系统, 能够跨越进程边界, 实现网络、语言、应用程序、开发工具和操作系统的"即插即用"。设计后的系统综合了VC++ 软件平台、软构件技术及农业系统的特点, 大大增强了其开放性和多技术的集成能力,

便于系统的理解和维护,为农业管理决策系统的开发提供了一种构件化的基础框架。

[参 考 文 献]

[1] 熊范纶,淮小永,丁立志. 基于软构件技术的新一代专家系统开发平台- 雄风 6.0[J]. 模式识别与人工智能, 1999, 12: 8~17.

[2] 潘爱民. COM 原理与应用[M]. 北京: 清华大学出版社, 1999.

[3] Zhu Yan, Cao Weixing, Luo Weihong, et al. Knowledge model and growth model-based intelligent decision support system for wheat management [A]. Proceedings of International Conference on Engineering & Technological Sciences[C]. Beijing: New World Press, 2000, 275~280.

[4] 李 旭,曹卫星,罗卫红. 小麦管理智能决策系统的设计与实现[J]. 南京农业大学学报, 1999, 22(3): 9~12.

[5] Lemmon H, Chuk N. Object-oriented design of cotton crop model[J]. Ecological Modeling, 1997, 94: 45~51.

[6] Kragtinski D J. Visual C++ 技术内幕[M]. 第 4 版. 北京: 清华大学出版社, 1999, 235~266.

[7] 马 丽,顾显跃,陶新峰. 小麦生长模拟模型软构件在 VC++ 中的实现[J]. 南京气象学院学报, 2000, 23(4): 601~607.

Application of soft component technology to design of intelligent decision-making system for crop management

Zhu Yan, Cao Weixing, Wang Shaohua, Pan Jie

(*Ministry of Agriculture Key Laboratory of Crop Growth Regulation, Nanjing Agricultural University, Nanjing* 210095, *China*)

Abstract: The knowledge model and growth model-based intelligent decision-making system for crop management integrates crop information and expert knowledge of multi-subject and correlative support technologies. Of key importance is to know how to design the functional modules of the system in order to make it convenient for a non-developer to understand and study the essential parts of the system, for functional modules to be reused and for the developer to maintain the system. The paper presents the selection of the software platform, the system structural design, the programming process of soft component with Visual C++ platform and key techniques; and thus provides a fundamental component frame for developing a decision-making system for crop management.

Key words: soft component technology; COM I/O; visual C++ software platform; intelligent decision-making system; crop management