# 面向框架的管理信息系统架构①

曾丽花[1], 柏东明[1], 杨蔼萱[2], 陈 靓[1], 郭晓东[1], 孙军军[3], 李让宽[4]

[1](中国石油勘探开发研究院西北分院 计算机技术研究所, 兰州 730020)
[2](北京航空航天大学 计算机科学与技术, 北京 100191)
[3](中国石油新疆油田 数据公司, 克拉玛依 834000)
[4](中国石油四川石化有限责任公司 设备检修部, 彭州 611930)

**摘 要**: 随着企业信息化建设的推进, 诸多管理信息系统的独立开发模式和不合理的框架设计耗费了大量的时间和人力. 研究和设计了面向框架的管理信息系统架构, 通过业务需求抽象和功能构件标准化, 缩短开发周期, 节省开发成本, 提高软件质量, 其可扩性和灵活性解决了相对固化的软件开发模式与企业快速变化的运营环境之间的矛盾. 该研究方案对管理及其他应用系统的研发均有很好的借鉴作用.

**关键词**: 面向框架; 重用; 灵活性; 可扩展性

## Applying Frameworks-Oriented Technology in Management Information System

ZENG Li Hua[1], BAI Dong-Ming[1], YANG Ai-Xuan[2], CHEN Liang[1], GUO Xiao-Dong[1], SUN Jun-Jun[3], LI Rang-Kuan[4]

[1](Research Institue of Petroleum Exploration & Development-Northwest, Petrochina, Lanzhou 730020, China)
[2](Computer Science Technology, Beijing University of Aeronautics & Astronautics, Beijing 100080, China)
[3](Data Company, Petrochina Xinjiang Oilfield Company, Kelamayi 834000, China)
[4](Department of Equipment, Petrochina Sichuan Petrochemical Co. Ltd, Sichuan University, Chengdu 611930, China)

**Abstract**: With Promotion of information technology in the Enterprise, independent development mode or unreasonable framework design of many management information system cost a lot of time and human resources. This article researches object-oriented frameworks technology in management information system, it reduces cycle of development, saves cost, improves quality of software, it's flexibility and scalability can adapt increasing and changing of information technology requirements very much. This technology can be reference resources of building office management system in large enterprise.

**Key words**: object-oriented frameworks; reuse; flexibility; scalability

在企业信息化进程当中, 管理信息系统在企业内部发挥愈发重要的作用, 业务逻辑和办公流程大量存在于这些系统当中. 随着信息化逐步深入, 系统与系统之间的通信和联动增多, 系统的独立开发模式, 导致系统设计模式差异加大, 系统接口建立和调用方式也各不相同, 而在已有的面向框架的系统设计中, 存在构件划分不准确, 功能边界不清晰, 耦合紧密等问题. 另外, 业务的频繁变化越来越要求软件系统必须具备扩展性和灵活性, 能够及时有效处理一些特殊和突发情况, 原有的构件设计和开发方式使得软件系统变更时间长、成本高, 也是阻碍信息技术在企业管理中更好应用的关键问题.

通过研究和应用面向框架的管理信息系统架构, 明确了系统构件的功能和关系, 缩减了系统开发周期, 缩小了系统间差异, 可灵活扩展的设计模式能够快速应对需求变更, 统一了系统的设计及开发标准和部署方式, 从而大大提高了软件可靠性和开发效率.

## 1 什么是面向框架技术

随着软件开发技术和工具的发展进步, 软件开发

① 收稿时间:2012-04-16;收到修改稿时间:2012-06-11

从面向过程到面向对象，再到现在的面向框架，不断地解决和适应现实软件的应用需要，面向框架是面向对象编程概念的扩展，面向对象编程是对客观对象用类的方法进行抽象，面向框架是针对企业的业务逻辑和业务流程进行抽象和归类. 与传统基于类的面向对象重用技术比较，应用框架更注重于面向专业领域的软件重用. 应用框架具有领域相关性，框架的粒度越大，其中包含的领域知识就会越完整. 应用框架是一个可复用、半成品的应用，框架预先设置了系统所需的主要框架构件及其相互联系，使开发者能够在此基础上定制自己的应用系统. 因此框架可以被许多其他应用所重用，框架具有模块性、可重用性与扩展性. 面向框架的编程方法有效地解决了基本定型的软件与企业运营环境变化间的矛盾，在提高软件开发效率、保障产品质量、降低开发维护成本方面具有较大优势.

## 2　面向框架研究

管理信息系统架构应贴近管理业务的需求特点和应用本质. 经过企业信息化过程当中的多个管理信息系统的需求调研、开发测试和最后的上线运行及推广运维，剥离系统中丰富的数据展现形式和多种多样的应用方式，从实际应用的角度去认识企业管理及办公业务，分析应用系统运行的本质特征和最终用户对系统的需求，可总结归纳出企业管理信息系统的应用实质主要包括以下五点：①存储和管理资源数据，定义业务应用的组织机构和用户信息；②实现与业务需求相对应的数据模型和存储格式；③构建符合企业管理要求的办公流程，使数据有序填充直至流程完结；④赋予并严格控制用户在流程环节当中对数据的读取和操作权限；⑤对系统内的业务数据，尤其对已经发布的数据，提供灵活丰富的查询检索功能.

### 2.1 框架设计

针对企业管理信息系统的应用实质，结合面向对象框架技术的思想，对应用框架的设计，首先面临的是领域业务逻辑的抽象、规划分类等问题，通过对该企业的业务逻辑和管理流程进行全面的综合分析，剥离出属于用户定义的数据，抽象归纳为应用框架所应具有的功能，整个管理信息系统应用框架应包含数据管理、工作流服务、系统管理三个主要框架构件.

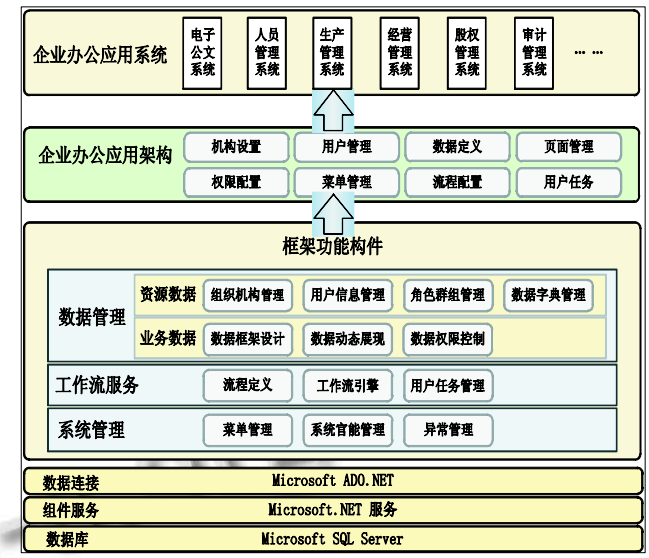


图 1　管理信息系统的应用框架

### 2.2 功能构件

#### 2.2.1 数据管理

数据管理构件主要包括资源数据管理、业务数据管理、权限管理等子模块.

(1) 资源数据管理模块负责组织机构、用户及角色信息的框架定义和基础功能. 组织机构信息主要以树型结构作为基础，以单根节点为原点向下无限延伸，并支持机构信息及其隶属关系自定义. 虽然，组织机构的数据结构较为简单，但在实际应用当中，必须对机构数据信息作以扩展，以便提供一些功能以满足一些企业内部特殊需求，例如灵活的机构信息更新和机构隶属关系变更功能以适应企业机构的频繁调整和变化；机构信息按业务板块、按级别等展示功能；为保证机构信息与历史数据的关联关系必须提供机构信息的逻辑删除功能；

用户和角色信息均与组织机构挂接，与其建立一对一关联关系；由于用户对数据的操作权限是随需求而灵活变化的，情况多种多样，而角色决定了对数据的操作权限，因此，用户与角色间应建立一对多关联关系，使系统能够定义用户对数据的复杂的授权操作；

(2) 业务数据管理模块主要通过对业务数据的抽象设计，实现业务数据的自定义功能，模块内置了实体数据和字段模板表，对用户自定义而建立的实体数据表和字段实现信息管理和统计功能，内容包括实体表表名、描述和数据字段名称、描述、数据类型、数

据长度及展现形式等. 在实体数据表被创建的同时, 组织机构与其建立一对多的关联关系. 另外, 在数据模型与用户交互时, 模块将根据数据所定义的展现形式, 动态加载对应的控件或者封装后的控件, 用以实现数据收集、显示及常用的数据校验和联动功能;

(3) 数据权限管理模块主要用于定义和控制数据条目权限和字段权限. 数据条目权限包括对数据条目的新增、修改、删除, 字段权限定义每条数据的每个字段的权限, 包括是否显示、只读、可编辑等. 数据权限管理模块向资源数据、业务数据和工作流服务构件读取数据人员角色、数据实体标识和字段信息及流程环节信息以决定数据操作权限.

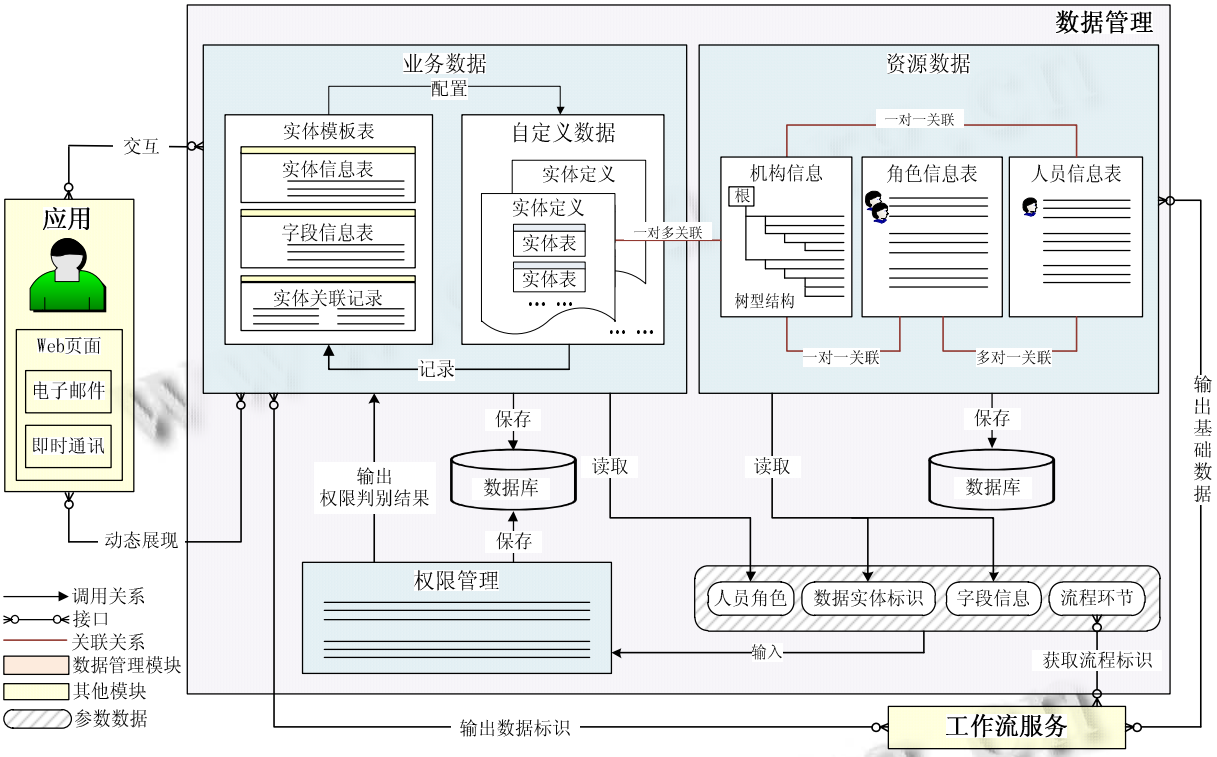


图 2 数据管理框架构件模型

2.2.2 工作流服务

工作流服务构件包括流程定义、工作流引擎、用户任务管理三个子模块.

(1) 流程定义管理, 负责把业务流程包含的数据编排成工作流引擎能够识别和工作的信息, 包括过程的开始和结束条件、组成活动、在活动间进行导航的规则、需执行的用户任务、所有工作流相关数据的定义等, 并建立和维护流程与资源数据中组织机构的关联关系; 工作流定义文件应采用 XML 语言描述, 提供了一种跨平台、跨网络、跨语言数据描述方式, 具备很强的可读性、易用性和可扩展性, 能够解决异构平台之间的无缝连接和互操作性问题;

(2) 工作流引擎, 以流程定义作为输入, 负责系统所有流程的有序流转、用户任务的分发和归并, 记录流程的层次和嵌套关系, 并保留流转痕迹及任务前驱和后继信息; 在较早的工作流服务设计和应用中, 往往是把环节作为流程流转的关键因素, 以环节作为流转的驱动, 也就是说, 驱动逻辑存在于每个环节当中, 如果出现新的流程流转需求, 需添加新的驱动组件与环节对应. 这样, 用户被限制在环节当中, 这种设计只适用于流转轨迹规则简单清晰的流程. 为提高引擎的灵活性和适应性, 工作流引擎将活动作为流程驱动, 将流转模式的颗粒度细化到用户, 环节只作为用户活动中具备标识作用的属性, 驱动逻辑被统一定义在活动入口内, 大大提升了对复杂流程的兼容性.

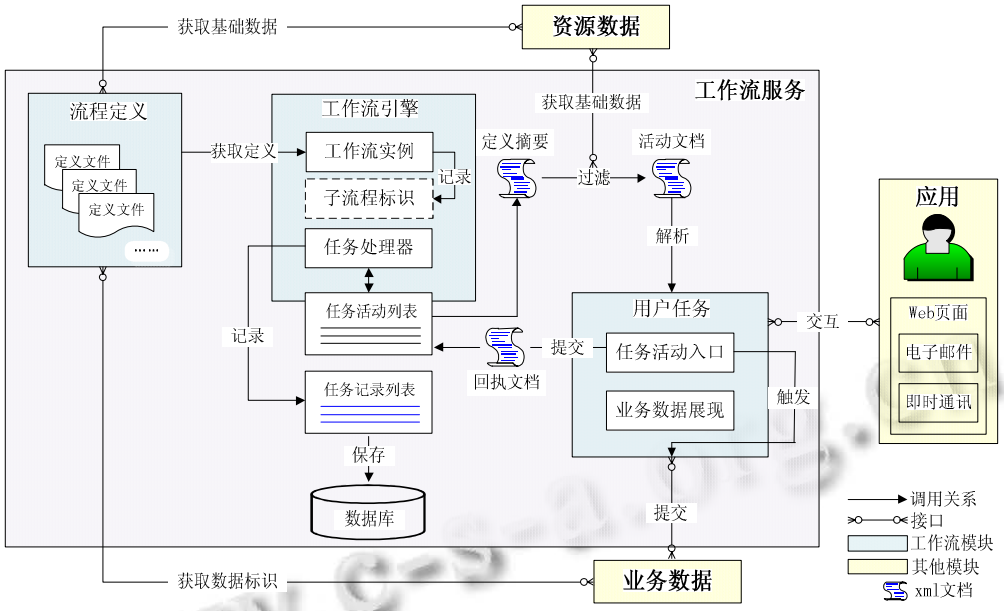


图 3　工作流服务构件模型

对流程的流转过程，工作流服务构件中建立了任务归并机制，归并方式应包括局部和全局归并两种.

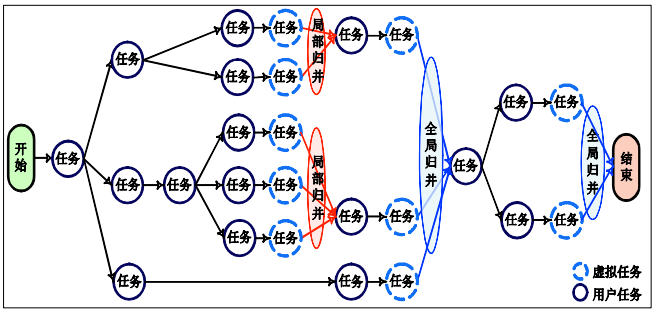


图 4　局部和全局归并

任务归并前由用户触发虚拟任务，它无需用户实际参与，只是标识了用户的流转意愿，归并的过程也是统计和分析虚拟任务的过程，并按照业务需求所对应的归并规则自动完成，如流转业务中常见的少数服从多数、多数服从少数、单一任务优先、任务数量优先等规则，这些规则也能够极大的提高流程的灵活性和适应性.

局部归并能够适应流程内某些环节的小范围并发，如部门内部发文的会签环节，而全局归并能够将流程所有分支聚合起来，满足流程整体归并和最终结束的要求.

(3) 用户任务管理，用于分配和管理用户任务，收集用户回执并提交给工作流引擎，触发流程流转. 用户进入任务后，模块申请由流程定义、流转信息和用户角色过滤后的活动文档，并将活动文档解析为可操作的活动入口； 用户提交任务后，把包括用户审阅意见和流转意图等信息的回执文档以 XML 格式提交给工作流引擎，同时触发业务数据更新;

2.2.3 系统管理

系统管理构件主要包括系统功能管理、菜单管理、异常管理三个子模块.

(1) 系统功能管理模块负责统计整理系统内的所有功能单元，判别用户对系统功能的访问权限. 界面是应用系统和用户间交互的最终形式，功能最终以界面或者界面组合的形式展现出来，每个功能单元对应多个用户界面，系统功能管理建立和维护了由界面组成功能、功能关联角色、角色对功能授权的对应关系，从而实现了界面信息编排、功能单元管理、访问控制的功能;

(2) 菜单管理模块，以用户所属角色过滤系统功能管理模块中所维护的界面信息，生成与用户权限相对应的系统菜单. 用户菜单内容中存放了系统内所有界面信息和与其对应的页面路径及参数，并与系统功能管理模块内的界面信息保持同步，将过滤后的菜单供系统为用户导航;

(3) 异常管理模块，任何应用程序被执行的过程

中都可能因为各种各样的原因引发异常，健壮的程序代码都会包涵异常处理的逻辑，便于快速定位和处理. 而框架模块作为系统基础服务，应更为重视对异常的记录和统计;

●模块内部异常，这类异常由模块内部或应用及扩展而触发. 例如，在工作流实例的生命周期中，流程定义与解释程序不匹配等而触发的异常； 在数据管理模块中，在继承抽象类的数据派生类内实现基类方法或接口，因缺少语句或返回结果不被识别而触发的异常;

●接口调用异常，这类异常发生在模块间或与其他系统的外部接口之间相互调用过程中，如接口调用函数不匹配、调用参数传递错误或因接口发布不可用等原因而产生;

●任务异常，这类异常主要是由于未通过应用数据的完整性和合法性校验、资源数据不可用、活动文档解析错误等原因而触发.

异常触发后，异常管理模块一方面应将异常信息和相关帮助信息提示给用户，另一方面，应记录异常的环境信息，如程序调用堆栈的完整异常信息(包括调用的函数或过程的名称、传递的参数名称和内容).

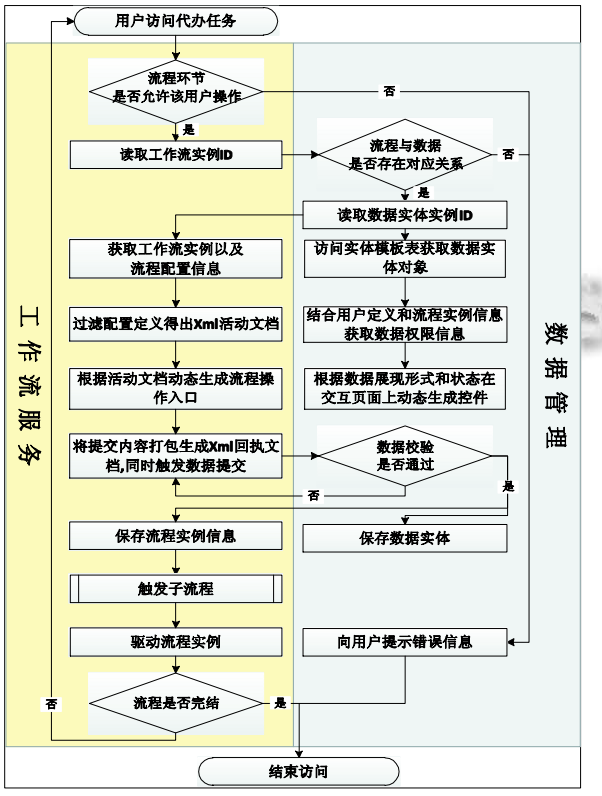


图 5 构件工作流程图

三个构件中，数据管理和工作流服务构件是框架内的核心，接口和调用较多，结合较为紧密.

数据模板是由在流程流转过程中逐步填充的，用户访问属于自己的代办任务参与流程，在完成任务的过程中填写相关数据，也就是说是由工作流服务驱动数据流转，为用户提供操作数据的入口.

用户通过访问 Url 进入代办任务，在校验任务归属正确性后，触发数据管理构件的校验和展现功能，两个构件使调用自身功能在 Web 页面上向用户动态生成活动入口和数据控件，完成相关操作后，工作流服务通过用户操作活动入口的事件触发数据管理构件的数据校验和保存功能，同时完成实例保存和驱动功能. 整个过程是工作流服务主动响应、数据管理构件被动调用的过程.

```
public class DataObject
{
   public int ID;                    //唯一性标识
   public Guid workflowID;           //工作流实例ID
   public DataEntities dataEntities; //实体集合
   // ... ...
}
public class DataEntities : List<DataEntity>
{ // ... ... }
public class DataEntity
{
   public int objectID;              //数据实体ID
   public int templateTableID;       //模板表ID
   public int fieldID;               //字段ID
   public string displayName;        //显示名称
   public object value;              //值
   public ValueType valueType;       //值类型(枚举)
   public ControlType controlType;   //控件类别(枚举)
   public ViewState viewState;       //显示方式(枚举)
   public string clientScript;       //客户端脚本
}
```

数据实体的类定义为 DataObject，类成员 workflowID 关联了其对应的工作流标识，成员 dataEntities 描述了数据条目集合体.

DataEntity 类定义了数据条目的关键成员，成员模板表标识 templateTableID、fieldID 和 displayName 间接实现了 O/R Mapping 映射技术，controlType 描述数据展现方式，ViewState 是根据权限判别得到的显示方式，value 和 valueType 将在用户提交后，保存所填充的值和值类型.

```
public class FlowInstance
{
   public Guid ID;                   // 唯一性标识
   public Guid parentID;             //父流程标识
   public int dataObjectID;          //数据实体ID
   public CfgDocument document;      //配置信息
   public string taskName;           //任务显示名称
   public TaskLogCollection taskLogs; //用户任务记录
   // ... ...
}
```

工作流实例类内同样通过 dataObjectID 记录了与其关联的数据实体 ID，CfgDocument 是流程配置文档 XML 格式反序列化后对应的类，构件内的活动和回执 Xml 文档均由此类序列化生成，TaskLogCollection 是工作流实例所分配的用户任务日志，用于记录用户操作流程的过程信息.

用户任务展现给最终用户的任务表单分为两个部分，一部分是由工作流服务将流程活动文档展现的活动入口，另一部分是由数据管理依据用户所选择的形式展现的业务数据控件.

### 2.3 设计模式

在框架构件中，根据其自身要提供的功能和应用特点，使用了多种设计模式，并形成了白盒和黑盒框架. 白盒框架，通过继承框架基类来扩展和实现接口，其中，广泛使用了 Template Method 模版方法，定义操作的算法骨架，将一些步骤延迟到子类中，使子类可以不改变一个算法的结构即可重定义该算法的某些特定步骤，这样能够对已有的功能进行复用和定制. 应用白盒框架，开发者必须对其内部结构有所了解； 而黑箱框架可以由实例化和配置的构件和类组成，允许对象经由组合和委托而被插入框中，通过定义遵循特定接口的类并使用像 Function Object 函数对象 Bridge Strategy 桥接策略，以及 Factory Method 工厂方法这样的模式来把这些类集成进框架中，可以对已有功能进行复用. 与白盒框架相比，黑盒框架可能更易于使用，因为开发人员无需对框架的内部结构有太多了解. 但是，黑盒框架也可能更难以设计，因为在设计过程中，必须定义能够预见到的一系列使用情况的接口.


图 6　用户任务表单

在框架中采用面向接口编程作为设计原则，用来隔离系统的业务变化，接口用于描述类的外部行为，被用来为不同的行为做建模；抽象类主要用于单个或多个接口的不完全的实现，定义实现构件中的公共行为；构件是以一个或多个接口构造的形式使用的，可以与其他类联合，是以指定接口和明确的上下文依赖组合的单元； 类则是框架的最低层.

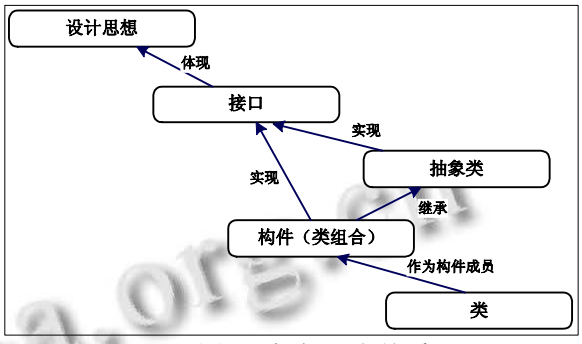


图 7　框架元素关系

### 2.4 采用技术

应用框架以 Microsoft.Net Framework(以下简称“Framework”)作为底层组件，使用 Microsoft Visual Stadio.Net 2010 集成开发环境(以下简称“VS.Net”)，采用 C#语言编写，数据库采用 SQL Server 2008.

(1) 在表示层，页面展现多为 ASPX 和 HTML 两种形式，为保持页面整体布局和统一风格，使用了主题和母版页技术，运用主题中的皮肤和级联样式表控制控件外观； 使用母版页创建一致的页面布局，即定义页面所需的外观和标准行为，并创建包含显示内容的内容页，这样，页面将以母版页的布局与内容页的内容组合在一起输出展现;

另外，为满足较好的用户体验，大量应用了 XML Document Object Model、JavaScript、XML Http、Ajax-JSON 等技术;

(2) 在逻辑层，主要利用接口技术，少量运用委托技术把业务中频繁变动的部分，也可以称为是类的行为从基类中隔离出来，极大增强的构件的适应性和灵活程度;

除使用公有类外，主要应用了抽象类和密封类，抽象类用于提供多个派生类可共享的基类的公共定义，通过创建派生类来提供自己的类实现； 为防止派生，应用了密封类，它不能用作基类;

为规范表示层到逻辑层、逻辑层到数据层的访问行为及功能调用安全性考虑，框架构件主要以程序集、类、派生类为访问边界，运用了类及其成员的 5 个级别的可访问性:①public:访问不受限制;②protected

－访问仅限于包含类或从包含类派生的类型;③Internal:访问仅限于当前程序集;④protected internal:访问仅限于当前程序集或从包含类派生的类型;⑤private: 访问仅限于包含类型;

应用泛型概念, 提升框架构件的可重用性, 在集合类、接口和参数中被大量使用;

应用反射技术, 动态创建类型的实例, 将类型绑定到现有对象, 或从现有对象获取类型并调用其方法或访问其字段和属性;

应用属性技术, 主要是 Framework 类库中的 Attribute 类和 AttributeTargets 类, 将预定义信息与诸如类、类成员、事件等目标元素相关联, 在运行时进行检查以控制程序处理数据的方式, 能够完成一些常规类型系统不直接支持的语言功能, 例如, 利用类成员的属性定义校验成员值是否合法;

应用序列化技术, 使用二进制序列化保持类型保真度, 即保留对象状态, 例如, 通过将对象序列化到页面的保持视图状态, 以起到缓存实例对象的作用; 使用 XML 序列化和反序列化, 结合属性技术, 很好的解决了使用 XmlDocument 类、调用类方法和校验的繁琐性, 大大提高了对 XML 文档的读取、编辑及生成保存等操作的便利性、准确性和规范性;

应用 Xml Web Service 技术实现外部接口, 并结合接口调用的频繁程度和交互数据大小等因素, 使用了集成 Windows 身份验证和基于 SSL 的证书两种验证方式以保证接口调用的安全性;

(3) 在数据层, 为保证数据库操作的完整性和数据一致性, 使用了数据库事务技术, 将每个业务执行过程所涉及的所有数据访问、编辑和更新等操作包含在同一个数据库事务内, 并对错误操作执行回滚;

在数据库函数或存储过程中, 应用了数据库的公用表表达式 CTE 新技术, 它能够引用其自身, 递归创建公用表, 重复执行初始公用表返回数据子集直到获取完整结果集, 递归公用表极大简化了查询所需的代码, 较之前的数据版本中, 使用临时表、游标和逻辑控制的执行效率也有较大提升.

## 3 结语

企业内现有的管理信息系统中或多或少存在着系统模块功能耦合紧密, 扩展难度大的问题, 难以适应企业业务的变化和特殊需求, 同时, 系统的独立开发模式导致系统在部署、接口调用方式、数据存储等诸多方面的多样化, 这些个性化问题增加了系统在开发、实施、对接、联动直到运维工作的复杂度, 而系统从无到有的开发过程也耗费了大量的时间、人力和财力.

面向框架的管理信息系统架构解决了这些问题:

●架构中独立的工作流服务构件设计使工作流服务成为业务数据操作的入口, 实现了业务数据模块与工作流服务的松散耦合, 清晰地划分了数据与流程的功能边界, 业务数据和流程环节的自定义功能把大部分软件变更工作推向用户前端;

●通过业务需求萃取和抽象得出的管理信息系统架构, 构件中内置了的基础构件, 采用可扩展的设计模式, 预留可扩展空间; 构件以黑盒模式面向系统实现, 使开发者有充裕的时间完成业务调研和需求分析, 而不必深陷底层模块的设计和开发;

●面向框架的系统架构不仅通过可重用的构件简化了系统的开发过程, 并且统一了数据存储格式, 规范了系统接口调用方式和输入参数及输出结果, 从而大大降低了软件生命周期内关键环节的工作量和工作难度.

## 参考文献

1 Horvat RV, Rozman I. Improving Object-Oriented Frameworks by Considering the Characteristics of Consti- tuent Elements, 2009,(1):1068－1075.

2 Fayad ME, Schmidt DC, Johnson RE(eds). Design of an object-oriented framework for measurement systems. Object-Oriented Application Frameworks, 1999,(9):20–25.

3 van Gurp J, Bosch J. Design, implementation and evolution of object-oriented frameworks: concepts and guidelines. 2001 (3):277–300.

4 Kim SG.一种基于构件管理模型的域框架设计.软件学报,2002,13(3):335–341.

5 范菊逸.企业级应用中面向对象框架的研究[硕士学位论文].武汉:华中科技大学,2005.

6 何昭,李传湘,崔巍.基于面向对象框架的软件开发方法.计算机工程,2002,28(4):5–7.

7 沈延森.快速可重构信息系统及其关键技术研究[博士学位论文].南京:南京航空航天大学,2001.

8 周警伟,罗晓沛.实施一个面向对象框架的方法.计算机仿真,2002,19(3):107–111.