

文章编号: 1005- 8893 (2004) 02- 0040- 04

动态 ERP 系统的设计与实现^{*}

周重益¹, 谢章丽², 宋刚正¹

(1. 江苏工业学院 计算机科学与工程系, 江苏 常州 213016; 2. 江苏工业学院)

摘要: 随着企业管理的日益发展, 各种企业管理系统也要求具备相应的支持。ERP 系统是企业管理的重要工具之一, 其动态性要求日益成为目前系统开发的热点。在框架和组件技术的基础上研究了动态ERP系统的构建方法。通过设计模式设计了一个动态可调的框架结构, 并通过框架对组件的调用实现了独立功能的易调整、易扩展特性。

关键词: 框架; 组件; 企业资源计划; 设计模式

中图分类号: TP 339

文献标识码: A

ERP 系统是目前企业管理的有效工具, 也是计算机应用的重要领域。随着世界经济的发展, 企业管理要求更加的高效和灵活, 以适应快节奏和多变的社会竞争环境。当前, ERP 系统在企业中担负起越来越重要的作用, 同时也面临着系统因无法适应实际管理需要而频繁升级的困扰。据统计, 目前国内企业中 ERP 系统实施的成功率还很低, 其主要原因是当业务流程需要改变时, ERP 系统不能很好地适应新的要求。EPR 软件是否能够成功, 很大程度上依赖系统对企业管理模式、管理方法持续改进的支持能力, 因而要求 EPR 系统必须满足动态性, 动态性成为现代 EPR 软件研究的焦点^[1]。国内外普遍认同的观点是构建一种可调整, 易扩展的动态 ERP 系统。笔者将这一理念与目前比较成熟的组件和框架技术相结合, 设计实现了一个基于中小型企业的动态 ERP 系统。

1 组件和框架技术简介

组件 (component) 是软件系统内可标识的、符合某种标准要求的独立程序体。组件具有语言无关性、接口标准化、可插用性等特点。基于组件技术的软件工程是目前软件工程领域的新课题, 其研究的内容就是通过组件这种“预制块”构建整个软

件系统的方法。主流的组件标准有 COM、CORBA 以及 EJB 等。

框架 (frame) 是由开发人员定制的应用系统骨架。它是整个系统或部分系统的可重用设计, 是一组相关组件的集合。框架可以分为应用程序框架、领域框架和支持框架。其中领域框架的使用十分普遍, 其特点是模块化、可复用性以及反向控制。框架通过预制一定的接口调用功能组件, 实现具体功能。因此通常可将同一领域中的共性特征抽象成为框架, 而具体功能通过组件加以实现。

2 系统设计

动态 ERP 系统的设计是针对具体应用领域进行设计开发的过程, 它大致经过了 6 个阶段: ① 领域分析, 确立领域的需求模型; ② 软件体系结构的开发, 确立应用领域中通用的结构模型; ③ 可重用组件的开发, 为系统提供通用功能的组件库; ④ 框架的设计, 提供具体系统所需的框架, 同时使该框架具备一定的通用性, 即使其具有设计复用性或实现复用性; ⑤ 具体功能组件的设计与实现, 提供系统所必须的而组件库中不存在的组件; ⑥ 框架的测试与系统实现, 经过反复测试的系统框架即可用于系统, 进而实现整个应用系统。

* 收稿日期: 2004- 04- 01

作者简介: 周重益 (1962-), 女, 江苏常州人, 本科。

2.1 系统总体设计

系统的总体设计首先应考虑其适用的领域, 通过领域分析可以获得系统的需求模型, 需要特别关注的是领域中通用的需求。在此基础上设计系统的体系结构, 使体系结构的设计体现领域特点。最后是将一部分通用的具体功能用通用组件的形式编译实现。经过测试后将其放入组件库中统一管理。

本系统体系结构的设计借鉴了目前 ERP 系统普遍采用的 3 层结构^[2], 即客户端、应用服务器层和数据库服务器层 (图 1)。其中客户端直接与用户交互, 采用图形用户接口 (GUI) 接收和输出数据。应用服务器层以框架为核心, 采用消息传递机制控制业务流程。数据库服务器层主要通过 DBMS 对各类数据进行有效管理。

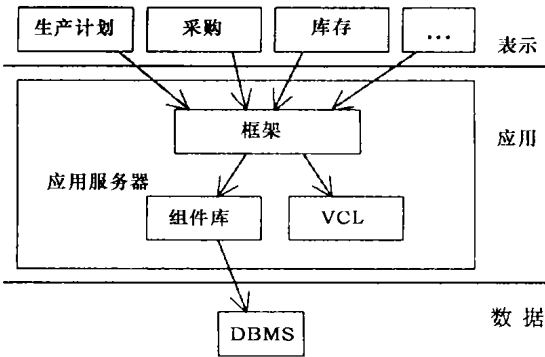


图 1 体系结构总体分布图

在体系结构设计中, 应用服务器层的框架设计是关键, 一个设计良好的框架可以充分体现系统的动态性。在本系统中, 我们采用继承机制实现了一个层次结构的面向对象框架。通常, 一个成熟的框架往往运用了多个设计模式^[3]。在本系统的框架设计中, 笔者也力图通过运用设计模式达到规范框架结构, 简化框架设计的目的。

2.2 框架的设计

动态 ERP 系统框架设计为 3 个层次: 公共对象层、业务处理层、调度层。公共对象层主要是一些可视化类的集合, 构成框架基本功能元素, 如文本框和一个显示树型结构的控件可以构成一个显示目录的功能元素。业务处理层在公共对象处理层的基础上体现业务功能, 它比功能元素粒度大, 主要由一些窗体体现。例如表单的录入、审核, 客户资料的修改等。分别向这些窗体传递不同的消息可以实现不同的功能。有时我们将业务处理层的窗体称为动态窗体或子框架。调度层是对业务处理的总体

流程控制, 在调度层设置热点 (Hot Spot) 可以让用户定义业务模块的分布, 从而根据用户要求设定企业的业务流程, 使框架具备动态性。调度层的功能主要有两方面, 一是提供用户定制系统结构的功能, 在业务流程需要变化时, 可以采用调度不同的子框架或组件实现业务流程重组。这一点是通过定义调度层控制类调整系统动态菜单实现的。二是调度层能够控制任意的子框架和组件实现相应的功能, 这一点需要调度层提供统一的接口调用组件或者是需要有一系列预先的约定向子框架传递消息^[4]。

整个框架的结构如图 2 所示。图中, 虚箭头为结构需要调整时, 框架各层允许进行的动态调度。

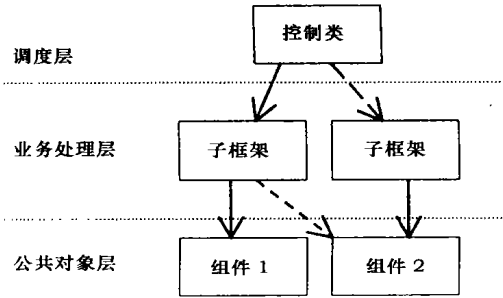


图 2 框架的层次结构图

2.3 组件的设计

通过现有的 ERP 系统以及领域工程和调研分析的结果, 可以将系统功能进行一个划分, 将具有独立功能特点、使用较多和重复的功能划分为组件。通常对于系统组件的提取, 我们有以下一些要求: ①具有较高的通用程度, 体现系统动态特点; ④具有统一接口, 使其易于组装; ④必须具有合适的描述机制, 便于检索复用。

组件的设计必须经过接口设计、功能实现、测试等阶段, 每个阶段都需要有详细的文档加以描述, 最终将组件及其工作手册一并存放于组件库中。

2.4 系统中使用的设计模式

2.4.1 状态模式

动态窗体的设计采用了 State (状态) 模式 (如图 3 所示), 该模式允许一个对象在其内部状态改变时改变它的行为。由于框架业务处理层的设计采用了统一的窗体样式, 每个窗体分别通过不同的内部状态实现不同功能。对应每个状态, 窗体可以调用不同的公共对象, 例如, 需要实现查询功能时

可以调用搜索对象。动态窗体通过调度层传递的消息设置其内部状态,从而确定具体功能。采用状态模式,使系统仅改变少数状态变量就能够实现业务功能的改变,减少了开发和维护的工作量。

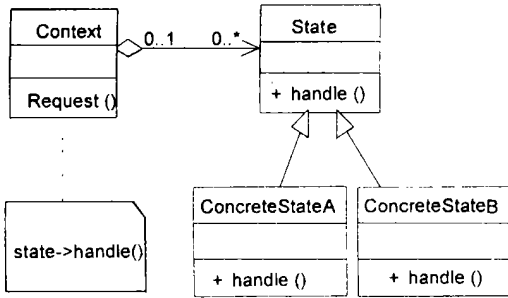


图 3 状态模式

2.4.2 访问者模式

动态菜单的菜单项对窗体的调用采用了 Visitor (访问者) 模式 (图 4), 该模式表示一个作用于某对象结构中的各元素的操作, 它可以在不改变各元素类的前提下定义作用于这些元素的新操作。将动态菜单作为接收访问者的元素类, 窗体作为访问者, 从而可以在不改变菜单类的前提下定义菜单新功能。在动态菜单中实现访问者模式必须在菜单项中预先定义窗体变量, 在窗体中预先定义状态变量。菜单项最终调用的功能由窗体变量和状态变量的值共同决定。采用访问者模式的优点是当需要增加一个菜单功能时, 无需改变动态菜单, 只需要增加一个窗体功能, 使菜单项和窗体之间实现动态绑定。

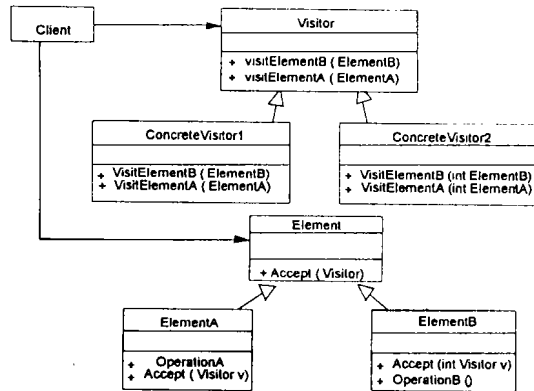


图 4 访问者模式

2.4.3 抽象工厂模式

抽象工厂类定义了一个创建产品的接口, 真正创建产品的是具体的工厂, 在框架的设计中主要是根据用户需求产生系统表单。抽象工厂实际上起到了表单模板的作用, 在其子类模板中使用工厂方法

(Factory Method) 分别生成表单的表头 (header), 表体 (detail) 以及脚注 (footer) (图 5)。用户可以选择参数, 调用抽象工厂模板就可以定制所需要的表单了。

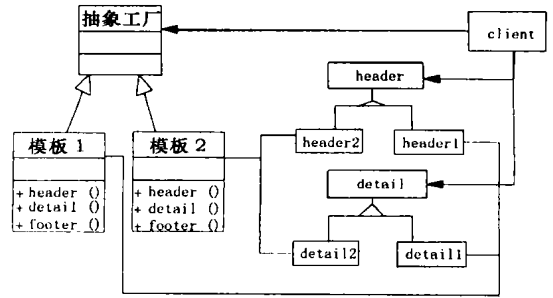


图 5 抽象工厂模式

3 系统实现

本系统使用 PowerBuilder (以下简称 PB) 作为开发工具, 通过设计进程内 COM 组件和动态框架实现系统动态功能。

3.1 组件的实现

在该系统中我们定义了若干组件, 如库存管理子系统通用查询组件, 库存管理组件, 用户信息管理等组件。

以库存管理组件为例, 我们定义了以下一些接口:

```

Iconnect () // 连接数据库
Idisconnect () 断开数据库
Iupdate () // 修改库存
Iinputbill () // 库存录入
Iretrieve () // 库存检索
    
```

接口的定义分别采用参数化和扩展点等方法, 以实现组件的复用性。定义完接口后, 在 PB 中实现各接口功能, 之后通过 PB 提供的组件向导实现对组件的编译和注册。可以将组件的工作手册用资源文件的方式绑定于组件。编写仿真环境, 调用组件, 测试其性能。经过测试后的组件可以统一存放于一个库文件 pbl 中。

3.2 框架的实现

首先定义若干公共对象层元素, 其中大部分可以通过 PB 自带的控件或定义用户类对象实现。由于每个 PB 控件和用户类对象都可以设置若干事件, 因此可以为一些通用事件预先设置代码, 使其子类继承时可以直接调用这些事件。

在业务处理层我们定义了多个窗体类型, 如查询窗体, 库存管理窗体, 打印预览窗体等。这些窗体都有多个状态, 分别将窗体类型及其多个状态记录在数据库中, 作为上层调用的依据。调度层的实现主要是控制类的实现。我们实现了控制类对动态菜单的显示和调用功能的控制。例如实现了对菜单名称, 显示顺序, 对应窗口以及窗口状态等功能的设置,

框架可以在调度层和业务处理层实现对组件的调用。通过在 PB 中设置 oleobject 类型变量, 用函数 connectToNewobject () 实现组件的调用。例如调用 PB80. query 组件可以作如下设置:

```
oleobject olequery
olequery. ConnectToNewobject ( " PB80.
query")
```

3.3 动态菜单的实现

我们在 PB 中预先定义好所有菜单项, 按一定的命名约定, 设置其编号, 并将菜单编号、名称、对应窗口类型、参数等信息记录于数据库, 作为调度层控制类修改的依据。设置完毕后将所有菜单项隐藏。系统运行后, 通过查询组件查询得到各菜单项的各属性, 将查询得到的菜单项置为显示状态, 并通过数据库中窗口状态参数传递相应的消息给动态窗体。

如果动态窗体需要改变, 只需通过调度层控制类将数据库中菜单项的对应参数加以修改, 便可在下次启动系统时由查询组件查询得到新的参数, 通

过其设置菜单项对应的窗体或窗体状态, 实现功能的转换。

4 结 语

经过测试, 本系统能够在很大程度上满足用户的动态需求, 尤其在业务流程改变时, 采用添加或修改组件以及改变框架消息的流向便可实现系统修改, 减少了系统维护的工作量。

动态 ERP 系统的研究力图改变原有 ERP 系统在结构和功能上的固定形式, 使其具有一定的可调整、可扩展机制。本系统的实现只是系统动态性的一个开始, 实现了一定程度的动态功能。由于开发环境、个人的知识结构等多方面的限制, 系统面向用户的动态功能还停留在高层次, 组件的动态调用仍需要程序员通过编写代码实现。希望通过今后的不断完善, 将开发级的动态性用更加直观的可视化方式体现出来。

参考文献:

- [1] 边世红, 薛劲松, 宋学艳. ERP 系统中持久对象动态性研究 [J]. 计算机集成制造系统——CIMS, 2003, 9 (5): 378–383.
- [2] 王凯, 石冰心. 三层体系结构在一个 ERP 系统中的应用 [J]. 计算机与现代化, 2002, 3 (4): 33–36.
- [3] Erich Gamma, Richard Helm, Ralph Johnson, et al. 设计模式: 可复用面向对象软件的基础 [M]. 李英军, 译. 北京: 机械工业出版社, 2000.
- [4] 万建成, 卢雷. 软件体系结构的原理、组成与应用 [M]. 北京: 科学出版社, 2002.

Designing and Implementation of a Dynamic ERP System

ZHOU Chong-yi¹, XIE Zhang-li², SONG Gang-zheng¹

(1. Department of Computer Science and Technology, Jiangsu Polytechnic University, Changzhou 213016, China)

Abstract: With the development of enterprise management, all kinds of enterprise management systems are required to possess corresponding support. ERP system is one important tool of enterprise management systems. Its dynamic capability has become the hot topic of current system development. This article makes a research of the method of building dynamic ERP systems based on framework and component technology. With the design patterns, a dynamic and adjustable framework is designed. And by calling components from the framework it realizes the facility of adjustment and extensibility.

Key words: frame; component; enterprise resource plan; design pattern