第 32 卷　第 19 期　　计　算　机　工　程　　2006 年 10 月
Vol.32　№ 19　　Computer Engineering　　October 2006
·工程应用技术与实现·　文章编号：1000—3428(2006)19—0233—03　文献标识码：A　中图分类号：TP277

远程抄表系统中实现模式的研究

张祖平，肖　波
（中南大学信息科学与工程学院，长沙 410083）

摘　要：电力行业的远程抄表系统中涉及的表计规约众多，传统的设计模式与开发方法难以满足应用需求。论文在介绍创建性模式概念和相关技术的基础上，结合远程抄表系统中实际应用的需求，从面向对象的设计思想出发，着重分析了工厂模式、单例模式和建造模式的特点，并从扩展性、系统开销大小和封装继承等方面对多规约处理问题提出几种模式融合的解决方案。

关键词：抽象工厂；单例模式；生成器模式；规约；软件重用

Research on Implementation Pattern of Telemeter Reading System

ZHANG Zuping, XIAO Bo
（School of Information Science and Engineering, Central South University, Changsha 410083）

【Abstract】 Based on creation pattern and requirements of the telemeter reading system, characters of factory pattern, singleton pattern and builder pattern are analyzed, and the thinking of OOD is introduced. This paper proposes the novel solutions of multi protocol with system expansibility, spending and encapsulation. The problem of the long-distance telemeter reading is solved completely and efficiently.

【Key words】 Factory pattern; Singleton Pattern; Builder pattern; Protocol; Software reuse

近几年日趋严重的新一轮缺电形势对电力的管理效率提出了严峻的挑战，而计算机及通信技术的迅速发展，特别是通用无线分组业务(General Packet Radio Service, GPRS)技术的发展和成熟为远程抄表系统的实施提供了良好的契机。

远程抄表系统的特点在于终端/计量点数量大，现场表计类型复杂，从管理的角度提出的需求是：对于尚未接入的电能表，系统应该能够做到只要用户提供电表通信规约和接口方式即可接入，因此远程抄表系统中的处理通信方式及规约的逻辑结构要有足够的通用性和灵活性，充分考虑系统扩展的要求，这样的特性使我们不得不从软件重用的角度出发来构建系统[1,2]。

在面向对象的设计中设计模式不断地被应用到软件工程的各个领域[4～6]。设计模式利用了对象的继承、组合和代理(delegation)，在比OOP高的层次上考虑问题。尤其是使用代理来对任何不稳定或不确定的方面，如状态、对象的创建、应用平台等进行封装，保证了源代码的重用和设计的稳定。

本文以电力行业中远程抄表系统为应用背景，在深入分析几种创建性模型应用于系统构架上的优劣的基础上，融合多种设计模式，提出了有效的针对多规约处理和扩展的系统设计方案。

1 远程抄表系统结构

远程抄表系统主要功能需求如下：

(1)将电表上显示的数值正确识别出来，并将该数值正确传送回去。

(2)主站直接抄读现场用户电表读数，可进行批量或个别选择抄读，自动保存抄读的历史数据。

(3)对抄收到的电表数据进行统计、计费、双地址储存，并形成详细的用电档案。

(4)可进行现场或远程用电校对。

(5)快速进行用电户用电量查询。

(6)分时段抄表，实现分时计费，能解决按电网负载的峰谷时段采用的峰谷电价的方式。

它的物理组成主要包括计量/监测设备，通信设备和应用系统 3 个部分。

在我们的系统中计量/监测设备主要包括：计量设备，采集终端，计算机网络设备；采集终端专门负责从计量设备采集数据并具有一定的储存容量。主站定时从终端收集数据并进行分析应用。

通信设备即计量设备与采集终端之间的 485 通道；终端与主站之间的通信通道，可以是光纤，电缆，也可以是 GPRS、CDMA 等无线通道。物理模型如图 1 所示。

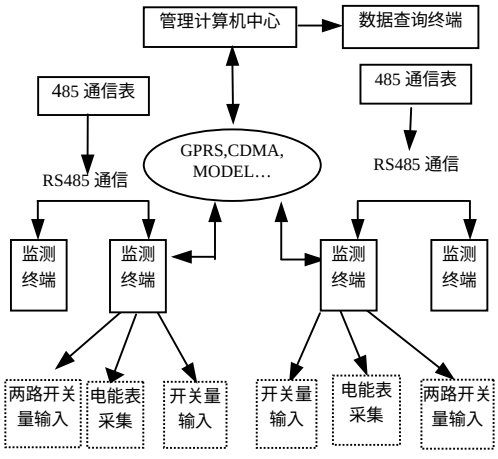

图 1　物理结构

根据上述物理结构，主站系统主要完成两大功能块：通信和应用。整个系统分成两大层，如图 2。

作者简介：张祖平(1966－)，男，博士、副教授，主研方向：数据库技术及应用，网络理论及优化等；肖　波，硕士生

收稿日期：2005-12-05　**E-mail：**zpzhang@mail.csu.edu.cn

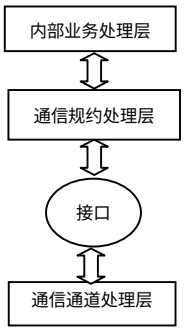

图 2 层次结构

(1)通信层：从低到高主要负责：1)通信通道的集成；2)多种规约的处理；3)通信参数处理，随抄补抄等通信方面的业务应用。

(2)应用层：主要负责数据应用和分析，从低到高主要负责：1)各类数据的汇总和计算；2)数据浏览显示；3)其它高级应用。

按照软件重用的设计思想，把功能相似的特性放到一个模块当中，模块与模块之间提供调用接口。当某一个功能需要进行扩充时，只须遵循接口规范，修改这一特定模块的代码即可。比如系统已经划分了规约处理模块与通信模块，并定义了它们之间的接口，当用户要求增加一种新的规约时，只须封装该规约的特性，并向通信层提供接口所统一的数据结构即可，无须根据各种不同通信通道的特性一一增加代码。

通信层和规约层完成的功能不一样，但基本设计思想是一样的，即提高系统的可维护和可复用性。为了达到这个目的，采用面向接口编程的方式，把业务逻辑都封装在内部，对外只传递和接收统一的数据结构。它们的示意用例图如图 3。

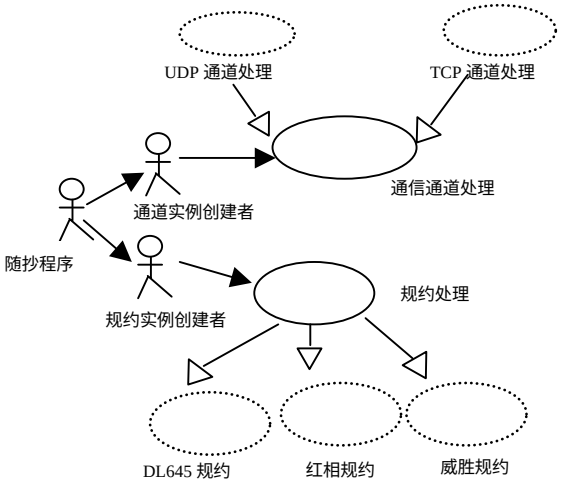

图 3 用例图

以下以处理多种规约为例说明如何在远程抄表系统中利用多种设计模式来实现系统易扩展的要求。

2 应用系统中涉及到的创建性模式

2.1 为什么需要创建型模式

电力行业的远程抄表系统中涉及的表计规约众多，如果沿用传统的创建对象的思维方式，则不可避免地将导致每集成一种表计规约必将从上到下修改程序的结果，这种设计严重违反了软件工程中易扩展的原则[1,6]，通过合理地引入多种模式和接口编程的思想[7]，这个问题就可以得到很好的解决。

若要创建一个 DL645 规约的对象，传统的方式是直接在程序中 new 一个，如 GBProtocol object＝new GBProtocol()；创建这个对象后，系统针对这个具体的 DL645 对象进行具体的组装、传送、接收、解析。当用户需要集成红相表的时候，开发人员又不得不修改程序，增加工作量：(1)创建对象：HXProtocol object＝new HXProtocol()；(2)组装层、通信层、解析层分别增加对红相表的处理逻辑。

这样的系统结构必将导致大幅度的程序修改，其产生的副作用是用户和开发商都承受不起的。

这就是需要在远程抄表系统中的多规约处理上使用多种创建模式的原因，然而采用哪种方式、如何构建是下面要重点讨论的问题。

(1)面向对象编程有一个重要的原则：面向接口进行编程，而不是实现。由于采用这一原则，在实际的编程时就使用了接口的概念，可是该接口一般有多个实现类，因此就会存在如何选择创建哪个实例的情况，如图 4。

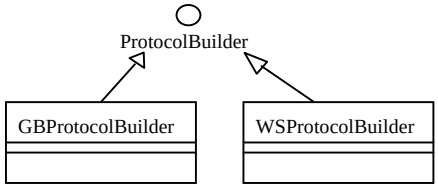

图 4 接口

传统的创建对象，不利于扩展、修改。如 ProtocolBuilder protocol=new GbprotocolBuilder()；这样一旦修改为其他的实现类(如生成一个威胜协议的实例)，需要多处修改代码，需要把创建对象外部化。

(2)如果采用形如“ProtocolBuilder protocol=new GBprotocolBuilder();”，那么每次创建都创造了一个新的对象，然而在电力行业的远程抄表系统，某一特定的表计协议处理仅需创建一个实例，在系统中来反复使用就可以了。

(3)一个产品常有不同的组成成分作为产品的部件，通常又叫做产品的内部表象。不同的产品可以有不同的内部表象。使用建造模式可以使客户端不需要知道所生成的产品对象有哪些零件，每个产品的对应零件彼此有何不同，以及怎样组成产品。在远程抄表系统中，一般上行下传的数据帧是由帧头，帧尾以及帧内容 3 个部分组成。对于通信层而言，它不需要知道各类协议中数据帧是如何形成的，只需要得到组装好的完整的数据帧对象即可。

用 3 个特征来归纳解决的方法：(1)创建对象外部化；(2)创建单一实例；(3)内部组装规约，提供统一帧对象。

2.2 使用 Factory 模式实现创建对象外部化

工厂模式专门负责将大量有共同接口的类实例化。工厂模式可以动态决定将哪一个类实例化，不必事先知道每次要实例化哪一个类。简单工厂如图 5 所示。

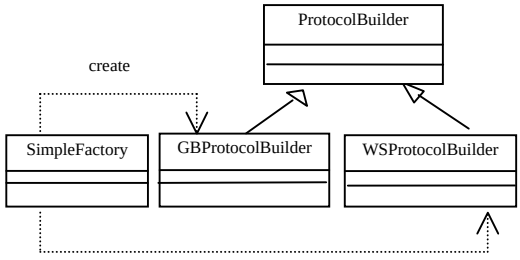

图 5 简单工厂

这种模式的核心是工厂类，这个类含有必要的判断逻辑，可以决定在什么时候创建哪一个产品类的实例。而客户端则

可以免除直接创建产品对象的责任，而仅仅负责使用产生的实例。如果在处理多规约的问题上采用这种方式，也可以实现把创建规约实例的工作外部化。然而各个地方的变电站所装挂的表计规约的种类繁多，如果采用这种模式，那么将来在远程抄表系统中需要扩充表计规约时就必须增加一个 ProtocolBuilder 的实现子类，并且要修改 SimpleFactory 的判断逻辑。这样一来必将破坏系统设计的开闭原则(一个软件实体应当对扩展开放，对修改关闭)。由此，在图 5 的基础上进行改进，使用形如图 6 的结构。

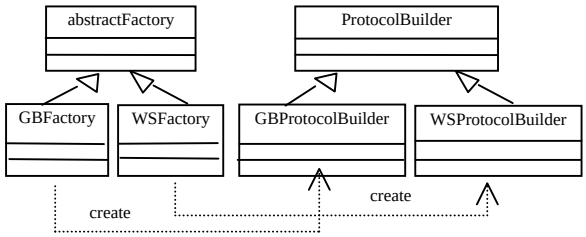

图 6 抽象工厂

这样一来核心的工厂类不再负责所有产品的创建，而将具体的创建的工作交给子类去做。核心类成了一个抽象工厂角色，仅负责给出具体工厂子类必须实现的接口。这样做的好处在于可以允许系统在不修改具体工厂角色的情况下引进新的产品。

采用这种模式就可以较好地解决上述的扩展问题，即当系统需要增加一种规约处理时，只需要增加相应的具体工厂类和具体 ProtocolBuilder 的实现子类，而不需要修改以前已经完成的代码。

2.3 使用 Singleton 模式创建单一实例

Singleton 模式主要作用是保证在应用程序中，一个类 Class 只有一个实例存在。使用 Singleton 的好处还在于可以节省内存，因为它限制了实例的个数。Singleton 模式具有 3 个显著的特点：(1)单例类只能有一个实例；(2)必须自己创建自己的实例；(3)必须给其他所有对象提供这一实例。

在远程抄表系统，某一特定的表计协议处理仅需创建一个实例，在系统中可反复使用。可以结合工厂模式中某一具体的工厂类来实现在系统中仅创建一个规约实现类的实例，类图如图 7。

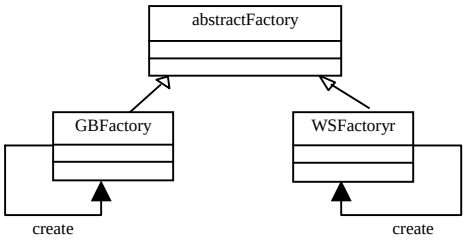

图 7 单例模式

2.4 使用 Builder 模式提供统一帧对象

在远程抄表系统多规约处理问题上，采用抽象工厂的解决办法，实现对多种表计规约处理的支持。新的问题是：必须处理在相同结构下，不同规约的各自组成办法。比如：一个规约的数据帧大致由帧头，帧尾以及帧内容 3 个部分组成，而我们希望看到不同规约各个部件的不同构成方式，这就如同建房子，用户希望看到同样结构下，不同风格房屋的外观。我们采用抽象工厂与 Builder 模式相结合的办法来解决这个问题：

(1)定义 ProtocolBuilder 类，即 inerface

在这个类中，对每一种协议组成元素都定义了一个 Builder 方法，并且定义了一个方法 GetFrame 返回一个数据帧对象。

(2)对每一种风格都定义一个具体的生成器(Concrete Builder)类,也就是具体规约的实现类：

GBProtocolBuilder: ProtocolBuider

WSProtocolBuilder: ProtocolBuider

…

并且在这些类中采用重载父类的方法。

3 多规约处理上的模式融合

电力行业的远程抄表系统有其特殊的特点：

(1)涉及的表计规约众多，在设计系统结构时必须充分考虑其扩展性和修改性。

(2)系统使用表计规约处理的地方很多，如随时抄表、定时抄表、主动召测数据、异常告警等功能都涉及到了规约的组成和解析。如果在每一个地方都创建一个实例，无疑给系统增加巨大的开销，如何保证系统中只有一个实例也是设计者必须考虑的问题。

(3)无论是哪种规约，数据帧一般都是由头、尾(校验和)和数据内容组成，然而对于用户来说只需要指定复杂对象的类型和内容就可以构建它们，用户不必知道内部的具体构建细节。

针对系统的这些特性，融合 3 种模式进行设计，类图如图 8。

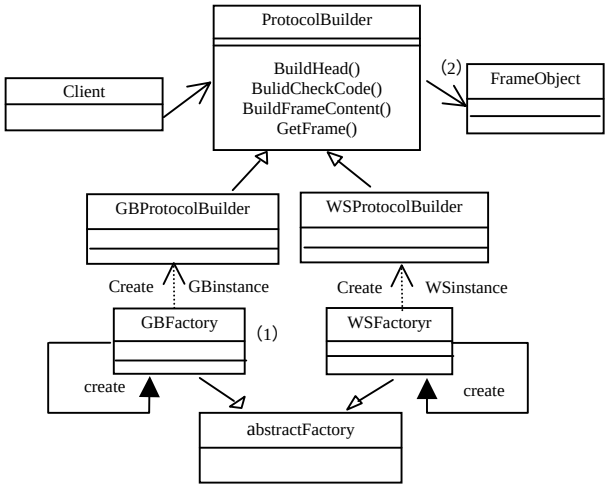

图 8 组合后的模式

图 8 中(1)所指即表示只产生一个实例的具体工厂；(2)所指即表示协议的生成器。

4 总结

针对远程抄表系统的具体需求，采用多种模式相结合的 Combination 模式，有效地解决远程抄表系统中多规约的处理问题，其最大优点是易扩展，即当增加一种规约时只需要增加具体工厂类和具体表计协议的实现类，而不必改变与通信层和应用层的接口；Combination 模式对某一特定的表计协议处理仅需创建一个实例，在系统中可反复使用，大大减少了系统开销；因此融合多种设计模式形成的 Combination 模式，是处理多规约的有效办法。在远程抄表系统中探讨实现模式的意义在于：

(1)保证每一个模块相对于其他模块独立存在，并只保持与其他模块的尽可能少的通信。这样一来，在其中某一个模块发生代码修改的时候，这个修改不会传递到其他模块。

（下转第 250 页）