设计模式在文件交换系统资源管理中的应用①

姚战伟[1]，孙 咏[2]，唐洪刚[3]

[1](中国科学院 研究生院，北京 100049)
[2](中国科学院 沈阳计算技术研究所，沈阳 110171)
[3](沈阳市安监局 安全生产应急救援指挥中心，沈阳 110034)

摘 要：从业务层的角度分析了传统资源管理系统在可扩展性和可维护性方面的不足，并借此提出了所要开发的资源管理系统。利用设计模式的思想，对复杂的业务层进行重新设计，引进了代理模式、简单工厂模式、访问者模式、双重分派和策略模式，使得文件交换系统功能模块组件和资源处理算法模块之间的交互关系间接化。业务层功能模块之间通过业务配置信息，调用相应的组件来完成一项业务，使每个业务组件之间不再发生直接联系，因此一个业务组件的更改不会影响到其它业务组件。采用这种开发方式基本上实现了开-闭原则(OCP)，降低了系统的耦合度，增加了系统模块的可复用性，构建了一个扩展性强、易于维护的业务系统。

关键词：设计模式；软件维护性；解耦合

Application of Design Patterns in Resource Management of the File Exchange System

YAO Zhan-Wei[1], SUN Yong[2], TANG Hong-Gang[3]

[1](Graduate University, Chinese Academy of Sciences, Beijing 100049, China)
[2](Shenyang Institute of Computing Technology, Chinese Academy of Sciences, Shenyang 110171, China)
[3](Shenyang Safety Supervision Bureau, Safety Emergency Rescue Command Center, Shenyang 110034, China)

Abstract: This article analyses the deficiencies of scalability and maintainability of the traditional resource management system from the perspective of the business layer of the system, and to put forward this paper is to develop resource management system. In this paper, the complexity of the business layer is to be re-designed through the design pattern idea, the introduction of a proxy pattern, a simple factory pattern, visitor pattern, double dispatch pattern and strategy pattern makes file exchange system the indirect interactions of function module components and resources processing algorithms. Functional modules of the business layer through configuration information call the appropriate components to complete a business, so that each business is no longer a direct link between the components, so a business component of the change will not affect the other business group pieces. With this development approach is basically to achieve the open - Closed Principle (OCP), reducing the system's coupling to increase the reusability of the system modules, and to build a scalable, easy to maintain business systems.

Key words: design patterns; software maintenance; decoupling

随着软件规模的膨胀，人们不仅关心程序的功能与效率，更加关心软件的后期维护。一个公认的事实就是：用户对软件功能的需求可能随时变化，而且这些变化不可能在软件开发期间全部知晓，这使得软件的后期维护变得更加困难。这就要求软件的设计者设计出稳定的软件，使得软件既能适应新的需求，又不必对软件做大量的修改，甚至不必修改，即设计的软件满足所谓的开-闭原则。

开-闭原则(OCP)的思想是：一个软件系统应当对扩展开放，对修改关闭。但是要想设计一个符合开-闭

① 收稿时间:2011-07-15;收到修改稿时间:2011-08-22

原则的软件系统并非易事，当然，也不可能有一种模式能用来设计所有的软件系统，因此，寻找一种或多种有一定应用范围的设计模式，使得按着这种模式或模式的组合而设计的软件系统能符合开-闭原则就变得很有意义。

国家水体污染控制与治理科技重大专项“辽河流域水环境风险评估与预警平台建设及示范研究”课题的数据交换平台的文件交换系统资源管理中，由于系统处理的文件数量众多，文件类型复杂，交换的数据量巨大，这些因素不仅增加了系统的复杂性，也不可避免的制约着系统的整体性能，因此采用合适的设计模式，对改善软件系统的适应性、灵活性，进一步提高软件系统的可维护性、可复用性就显得非常有价值。

1 文件交换资源管理系统概述

在传统文件交换系统资源管理中，系统的基本实现流程是按文件资源上传前的资源内容审查及文件类型过滤和处理，完成以上步骤后把资源上传到服务器端数据库或硬盘，然后在用户需要时，可以通过客户端进行文件资源管理，其中包括删除、修改、文件片段预览、文件全文显示或文件下载等。其用例图如下图 1 所示：

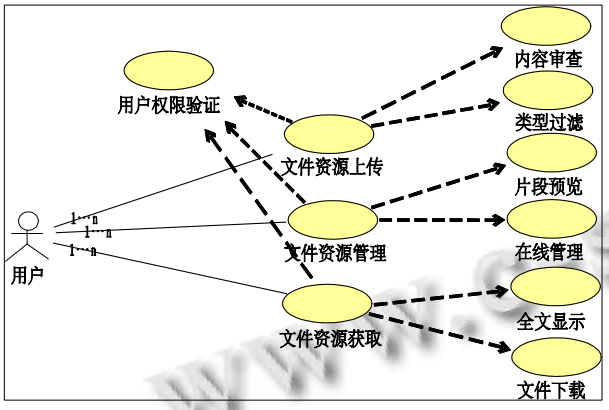

图 1 传统文件交换资源管理用例图

传统文件交换系统资源管理(如图 1)的软件架构大多数采用典型的 MVC 开发模式。这在一定程度上使软件系统的业务模块、显示模块和控制模块相互分离，降低了软件系统的耦合度。在面向对象的软件工程中，耦合是对一个软件结构内不同模块之间互连程度的度量。虽然 MVC 开发模式的采用在一定程度上降低了整个系统的耦合度，但资源管理的业务层中各业务之间，仍然存在着紧密的耦合关系，如果一个业务模块有所改变，就有可能会导致多个模块的改变。

本文所要开发的文件交换系统资源管理部分对业务层重新设计，采用代理模式、简单工厂模式、访问者模式、双重分派、策略模式，实现了业务层功能组件和资源处理算法模块之间相互分离，使得功能组件的增加或版本升级，或资源处理算法的减少、增加或修改，不会对其它模块或系统其它部分产生任何影响，满足了软件设计“对修改关闭，对扩展开放” 的基本准则，这在一定程度上实现了 OCP 原则。其用例图如下图 2 所示：

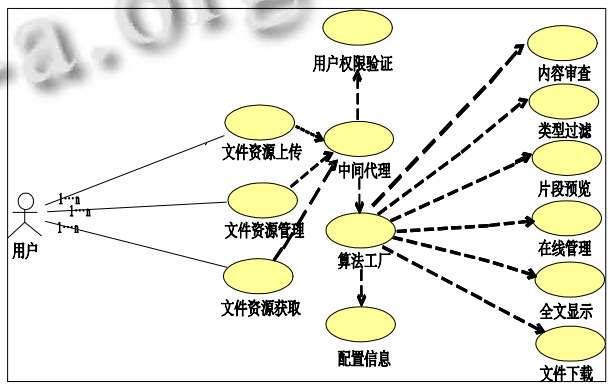

图 2 本文设计开发的资源管理系统用例图

2 设计模式应用

基于以上特点，本系统综合采用了代理模式、简单工厂模式、访问者模式、双重分派、策略模式。系统功能模块的扩展、文件处理算法模块的删除、增加或修改，对系统其它模块或功能组件几乎毫无影响，有效的降低了模块之间的交互耦合。并且由于功能模块和算法模块的独立，使得以后对系统的扩展和系统功能模块的复用都大有益处。

2.1 代理模式和简单工厂模式的应用

代理模式是对象结构模式。代理模式给某一个对象提供一个代理对象，并由代理对象控制对原对象的引用。在一些情况下，一个客户不能直接引用一个对象，而代理对象可以在客户端和目标对象之间起到中介的作用。

简单工厂模式是类的创建模式，是由一个工厂对象决定创建出哪一种产品类的实例。

本文通过在文件资源管理功能组件和资源处理算法模块之间的交互中引入代理模式和简单工厂模式，使组件和算法之间的耦合解脱开来，组件和算法的独立，

使得资源处理算法的增减、修改都不会对客户端和组件产生影响。

以获取文件资源为例。用例图如图 3 所示，当客户端获取文件资源时，①首先实例化文件资源获取功能组件和代理类。②代理对象通过用户权限验证模块来验证用户是否有权限访问相应资源。③代理对象根据功能组件传入的编号，调用工厂对象通过配置信息（如图 4 所示）获得相对应的资源处理算法对象类，通过反射机制得到算法对象，然后代理对象调用组件对象的 doWork()方法，把算法对象作为参数传入。④组件对象的 doWork()方法执行传入的算法对象的 doWork()方法以及其它逻辑，完成相应的资源获取功能。代理模式和简单工厂模式应用的顺序图如下图 5 所示。

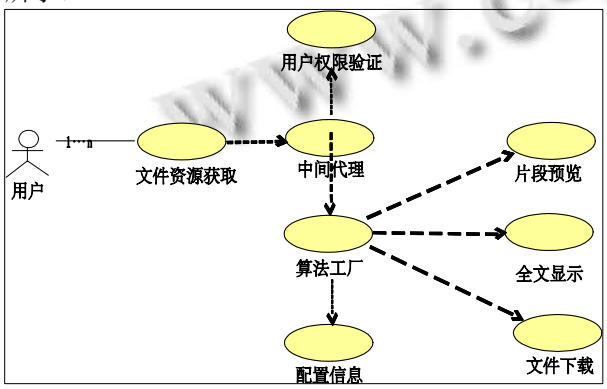

图 3　文件资源获取用例图

```
<?XML   Verson "1.0" en=oding "UTF-9"?>
<component-get>
  <compoent-get-name>fLLe-cesouce-manage</compoent-set-name>
  <component>
    <compoent-number>ra882459</compoent-number>
    <compoent-class>
      an.cn.siot.etl.compoent.resouceAcquisitioncompoent
    </compoent-class>
    </handle-alqorlthm>
      an.cn.siot.etl.algormlehm.FullTexcshowALg
    </handle-alqorlthm>
    <handle-alqorlthm>
      an.cn.siot.etl.algorithm.FileDownloadAlg
    </handle-alqorlthm>
  </component>
  <component>
    <compoent-number>ra489236</compoent-number>
    <compoent-class>
      an.cn.siot.etl.compoent.FIleUplload
    </compoent-class>
    <handle-alqorlthm>
      an.cn.sict.etl.algorithm.contentcheckAlg
    </handle-alqorlthm>
    <handle-alqorlthm>
      an.cn.sict.etl.algorithm.TypeFilterAlg
    </handle-alqorlthm>
    </compoent>
```

图 4　配置信息

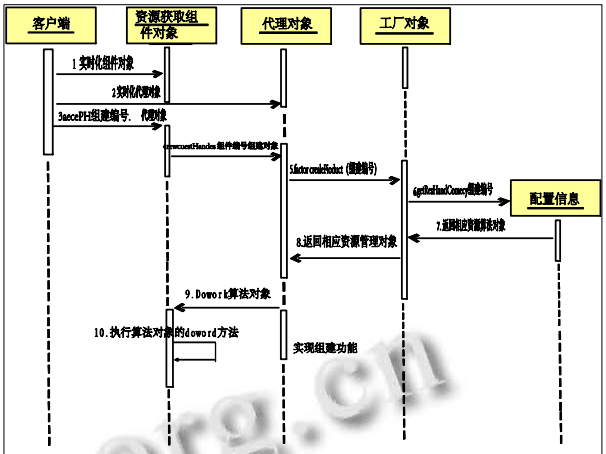

图 5　代理模式和简单工厂模式应用顺序图

传统文件交换资源管理系统功能组件的增减和修改困难的原因是:功能组件和文件资源处理算法逻辑紧密的耦合在一起，算法逻辑的修改必然会导致功能组件逻辑的变化，这样的系统很难实现软件模块和算法模块的复用并且系统的可维护性也很差。本文在引入代理模式和简单工厂模式后，功能组件的增加只需要实现功能组件接口，并且给新增加的功能组件分配相应组件编号，在配置信息里面配置该组件编号相对应的文件资源算法的类名即可。这样功能组件的增加不会对其它组件造成影响，增加了软件的可维护性和健壮性，在一定程度上满足“对修改关闭，对扩展开放”软件设计的基本原则。

2.2 访问者模式和双重分派的应用

访问者模式的目的是封装一些施加于某种数据结构元素之上的操作。一旦这些操作需要修改的话，接受这个操作的数据结构则可以保持不变。访问者模式适用于数据结构相对未定的系统，它把数据结构和作用于结构上的操作之间的耦合解脱开，使得操作集合可以相对独立地演化。在此模式里，数据结构的每一个节点都可以接受一个访问者的调用，此节点向访问者对象传入节点对象，而访问者对象则反过来执行结点对象的操作。这样的过程叫做“双重分派”。节点调用访问者，将它自己传入，访问者则将某算法针对此节点执行。

访问者模式涉及到二个角色，一个是访问者，一个是节点。在本文所开发的资源管理系统中，把功能组件类作为节点类，而把中间代理类作为访问者类。以文件资源获取为例，访问者模式应用类图如图 6 所

示，顺序图如图 7 所示。

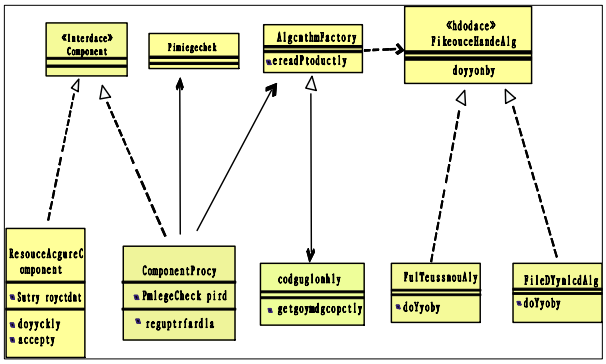

图 6 文件资源获取类图

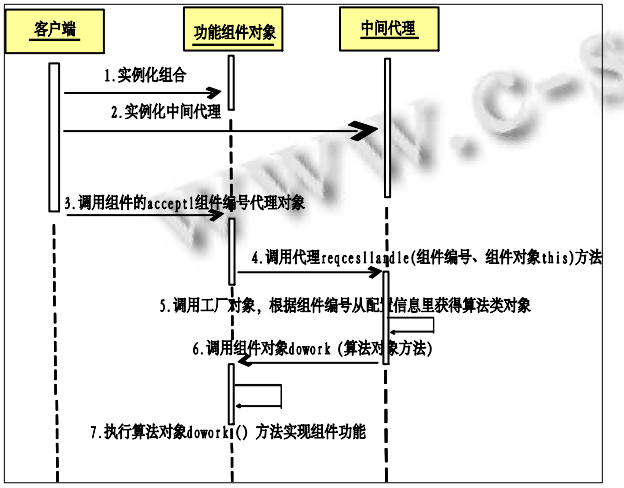

图 7 访问者模式应用顺序图

这样的结构使得功能组件和作用在功能组件之上的处理算法操作分离开来，并且很好的解决了功能组件结构不定，有时需要经常修改的问题。由于把功能组件和作用在组件之上的操作之间的耦合解脱开，这样功能组件的增加和修改，对处理算法不会产生任何影响，在一定程度上满足了开-闭原则，增加了系统的可维护性和可复用性。

2.3 策略模式的应用

策略模式属于对象的行为模式，其用意是针对一组算法，将每一个算法封装到具有共同接口的独立的类中，从而使得它们可以相互替换。策略模式使得算法可以在不影响到客户端的情况下发生变化，同时也可以动态地让一个对象在许多行为中选择一种行为。

如下图 8 所示，本文所要实现的文件交换系统的资源管理模块中，文件处理算法有 6 种之多，并且在系统的各个部分都有可能用到相同的算法，为了使文件处理算法可复用，对算法进行封装是必要的。

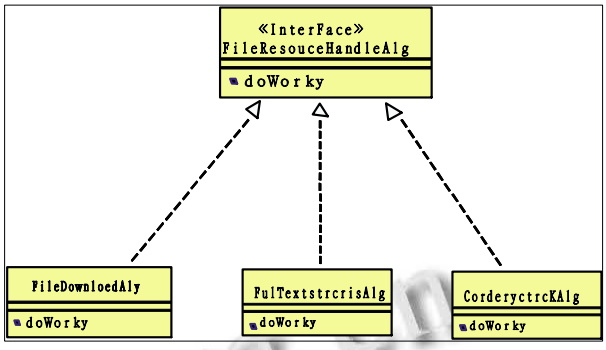

图 8 访问者模式应用类图

本系统中，由于文件数量众多，文件类型复杂，交换的文件数据量巨大，在一定程序上导致文件处理算法类过多。策略模式的使用很好的解决了系统算法类过多，难以维护的缺点，同时避免了让客户端涉及到不必要接触到的复杂的和只与算法有关的数据，另外算法处理模块的增加和修改对客户端来说完全透明。

3 结语

诸多设计模式的使用，使得本系统的可维护性和可复用性得到了很大提高。以下是采用设计模式后，系统的部分优点：

1) 封装了较为完善的文件处理算法，从数据层到业务层都应用了比较合理的设计准则，很容易作为子系统和其它系统集成。

2) 大降低了类与类之间的耦合度，很多算法和逻辑可以单独提取，作为类似系统深入扩展的基础。

3) 解决了将来可能资源管理模块的增加、修改或删减等变化可能导致的问题。本文采用简单工厂模式和代理模式使得资源管理模块的变动对系统的改动程度降到了最低。当需要增加资源管理模块时，只要根据要求，实现相应的接口即可，上层的业务代码不会有任何变化。

4) 能够很好的适应算法的变动。算法是本系统的核心，为了适应用户需求的变化，算法可能会随时变动。为了适应上述变化，本系统在算法实现过程中使得了策略设计模式，来协调不同的算法。这样系统已经达到了足够的灵活性。

(下转第 35 页)

响应最短的资源节点承载。

其中，F 表示用户需求特性权重，LT 表示服务器响应时间，μ 表示时间优先权转换比。当用户请求到 $\Pr iority = Unit_{total} \times F + LT \times \mu$ 达时，Priority 值动态更新。

最后云平台会跟根据运行在每个节点上的 Agent 反馈的状态信息对每个服务器节点进行实时监控，一旦某个节点发生故障，云平台会将用户桌面迁移至其他资源节点。

3 结语

云计算对整个 IT 产业的影响日益深远，它已成为运营商解决自身 IT 问题的重要手段。云计算已经不再局限于一种技术手段，它开始更多的从服务模式、产业链构成、管控架构方面改变企业。桌面云作为云计算破坏式创新的产物从终端管理薄弱环节入手，优化 IT 产品和服务，并推动传统 IT 运营体系的变革。本文搭建起一套适应未来发展的一体化桌面云架构，并在此架构下，结合运营商营业厅类生产任务型场景，通过引入业务重要性级别对 hypervisor 层资源调度算法进行优化，同时在此基础上对虚拟资源调度策略进行改进，以提升用户体验。本文所设计的架构与优化策略已成功应用于国内某运营商，有效推动了其终端云化进程，解决了终端管理面临的种种难题。

参考文献

1 杨林凤.基于 XEN 网络虚拟化的性能研究.上海:复旦大学,2010.40–47.

2 顾振宇,张申生,李晓勇.Xen 中 Credit 调度算法的优化.微型电脑应用,2009.3–5.

3 肖斐.虚拟化云计算中资源管理的研究与实现.西安:西安电子科技大学,2010.30–43.

4 王新佳,田晨,熊桂喜.基于多级队列算法的 ITS 资源调度策略.计算机工程,2003:4–5.

5 董耀祖.基于 x86 架构的系统虚拟机技术与应用.上海:上海交通大学,2006.50–56.

6 鲁松.计算机虚拟化技术及应用.北京:机械工业出版社,2008.78–85.

7 周铁成.虚拟化技术在数据中心架构中的应用研究.现代计算机,2009:3–4.

（上接第 17 页）

参考文献

1 孙咏.基于 OCP 软件应用架构的设计与实现.北京:中国科学院研究生院,2009.

2 阎宏.JAVA 与模式.北京:电子工业出版社,2002.41–44.

3 Shalloway A.熊节译.设计模式精解.北京:清华大学出版社,2004.

4 Gamma E, Helm R, Johnson R.李英军译.可复用面向对象软件的基础.北京:机械工业出版社,2000.

5 Martin J, McClure Carma.软件维护---问题与解答.北京:机械工业出版社,1990.21–39.

6 邹娟,田玉敏.软件设计模式的选择与实现.计算机工程,2004,30(10):79–81.

7 Shalloway A, James R.设计模式解析.徐言声译.北京:人民邮电出版社,2007.

8 Gamma E.设计模式--可复用面向对象软件基础(中译本).北京:机械工业出版社,2000.1–21.

9 张跃平.符合开-闭原则的一种设计模式及应用.大连交通大学学报.2007,28(1):38–40.

10 Shalloway A. Design patterns explained.vPearson Education, 2002.