非现场监测系统的应用研究

To Build on long – distance inspect application system

郭 宁 （首都经贸大学 信息学院 100026）

摘要：面对日益复杂的软件系统，要达到软件产业发展所需要的软件生产率和质量，软件复用是一条现实可行的途径。建立一个应用系统需要重用很多已有的组件模块。在这种情况下，应用系统的开发过程就变成对组件接口、组件上下文以及框架环境一致性的逐渐探索过程。

关键词：可复用技术 软件框架 组件

1 引言

为了避免软件系统开发中经常出现的开发效率低、费用高、系统难以维护，质量难以保证，不能按时完成系统的开发任务等问题。在开发银行非现场监测系统中，通过采用了面向对象技术、构造可复用的“组件”、选择成熟的软件设计模式等策略，在一定程度上使该软件系统提高了开发效率和质量，从而为整个金融监管综合信息系统的开发打下良好的基础，积累了实践经验。同时，也为推广实施这一全新的开发思想和技术提供了一种可行的技术解决方案。

2 软件可复用技术

软件复用就是将已有的软件成分用于构造新的软件系统。可以被复用的软件成分一般称作可复用组件，无论对可复用组件原封不动地使用，还是作适当的修改后再使用，只要是用来构造新软件，则都可称作复用。软件复用不仅仅是对程序的复用，它还包括对软件生产过程中任何活动所产生的制成品的复用，如项目计划、可行性报告、需求定义、分析模型、设计模型、详细说明、源程序、测试用例等。较高级别的复用容易带动较低级别的复用，因而复用的级别越高，可得到的回报也越大。

组件是可复用的软件组成成分，可被用来构造其他软件。它可以是被封装的对象类、一些功能模块、软件构架、设计模式等。对象按规定经过适当的接口包装之后成为组件，一个组件通常是多个对象的集合体。从广义上讲，以嵌入后马上可以使用的“即插即用”型组件概念为中心，通过组件的组合来建立应用的技术体系。狭义上讲，它是通过组件组合支持应用的开发环境和系统的总称。软件组件的特点是：其构成粒度大小自由，便于扩展；通过规定一个统一标准，建立起系统之间的智能互操作机制和语言独立性；外界仅通过接口访问组件；多侧面性，即组件表达的语义层次高，可以从不同侧面进行连接，外部特性不唯一；支持封装、继承、多态性。

构架是整个或部分系统的可重用设计，表现为一组抽象组件及组件实例间交互的方法。可以说，一个构架是一个可复用的设计组件，它规定了应用的体系结构，阐明了整个设计、协作组件之间的依赖关系、责任分配和控制流程，表现为一组抽象类以及其实例之间协作的方法，它为组件复用提供了上下文关系。因此组件库的大粒度重用也需要构架。按功能划分，可分为三层：基础层为基本数据类组件和系统支撑组件；中间层为各种通用的中间件；顶层为针对各种领域的专用组件或子系统组件。

组件通常是代码重用，而设计模式是设计重用，构架则介于两者之间，部分代码重用，部分设计重用，有时分析也可重用。在软件生产中有三种级别的重用：内部重用，即在同一应用中能公共使用的抽象块；代码重用，即将通用模块组合成库或工具集，以便在多个应用和领域都能使用；应用构架的重用，即为专用领域提供通用的或现成的基础结构，以获得最高级别的重用性。

3 非现场监测系统领域分析

非现场监测系统是银行监管部门对各级金融机构报送的数据、报表和有关资料，以及通过其他渠道（如媒体、定期会谈等）取得的信息，根据《巴塞尔协议》的有关规定对各种指标数据进行电子化处理，进行加工和综合分析，并通过一系列风险监测和评价指标，对金融机构的经营风险做出初步评价和早期预警的系统。例如：对单个银行，进行资本充足性及资本金变化情况、盈利的变化情况、市场风险变化情况、资产规模的增减变化和结构调整情况等进行监测分析；对银行整体，进行变化最大的银行业务领域、最可能出问题的机

构和业务等进行监测分析。

软件重用可区别为横向和纵向重用。横向重用是指重用不同应用领域中的软件元素,例如数据结构、分类算法、人机界面组件等。标准函数库是一种典型的、原始的横向重用机制。纵向重用是指在一类具有较多公共性的应用领域之间进行软组件重用。因为在两个截然不同的应用领域之间实施软件重用的潜力不大,所以纵向重用才广受瞩目,并成为软件复用技术的真正所在。不难理解纵向重用活动的主要关键点即是领域分析,根据应用领域的特征及相似性预测软组件的可重用性。一旦根据领域确认了软组件的重用价值,便可进行软组件的开发并对具有重用价值的软组件进行一般化,以便它们能够适应新的类似的应用领域。可复用软件系统开发过程如图1所示:

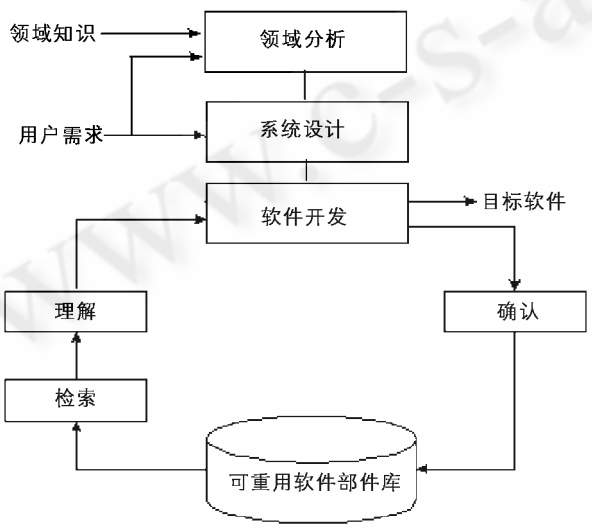

图 1　可复用软件开发过程

应用软件的开发过程是通过对特定应用领域的业务分析得到领域模型,由领域模型抽取出软件体系结构,软件体系结构通过参考设计模式抽取系统共性得到系统构架(Framework),Framework 中添加由业务领域中参考设计模式抽取出的组件,形成应用软件系统。软件体系结构、Framework、组件等部分成为应用软件复用的基础。

领域分析的任务是针对单个或一类相似的领域,以软件重用为目标,探寻并挖掘领域或领域族中能够为多个目标软件系统共用的软组件,并对它们进行结构化组织以备重用。领域分析活动不同于通常对特定系统进行的需求分析,它是对特定应用领域中已有的系统、预期的需求变化和技术演化进行分析,目的是标识出整个领域中通用的构架和相同的功能与接口。领域分析的结果将影响到系统需求的取舍,由此构造出的系统由于更适应变化的需求,日后被复用的可能性也更大。

在非现场监管系统领域分析过程中可大致归纳以下步骤:首先是寻求监管业务共性,形成监管业务的领域模型;其次是将公用的可复用部分和可变部分分开,然后建立分层的软件体系结构;第三是发现并描述可重用的实体,对这些实体及它们之间的关系进行抽象化、一般化和参数化,形成层次型的组件库,每一层的组件都将用到下一层组件的属性和方法;在开发具体应用系统时,仅仅修改可变部分和进行系统动态集成,形成应用软件系统。

领域分析必须进行一般化、抽象化和参数化,以抽象后的领域模型元素表示同类领域中不同软件项目之间的相似性,通过参数实例化刻画差异性,从而实现领域模型元素面向不同软件项目的可适应性和灵活性。非现场监测系统是采用基于 J2EE 规范的多层构架。如图2所示,以用户管理组件里新建用户为例说明了业务组件是如何在 Web 层和 EJB 层遵循和实现构架的设计机制。

其中图上半部分的 WEB 应用构架是抽象后的通用机制,而图下方的部分则是根据用户管理的需求开发的应用代码。

4　组件设计和抽取

由于组件表示一个或多个较细粒度类的逻辑集合,且被定义在较高级别,具有较粗粒度级,所以一个组件可以直接封装一个已划分的问题,该问题更直接地产生于分析与设计模型。一个更快地企业开发过程可以通过把服务(功能)直接的映射为更粗粒度的组件来实现。组件模型描述了系统中的组件层次。通过该模型可以清晰的看到系统组件之间的接口、静态关系和它们的交互关系。组件为组件用户提供多个接口。接口封装了组件提供的服务,隐藏了实现细节的可见性。非现场监测系统高层的组件结构如图3所示:

商业逻辑服务(Business Engine Services)由 FrameWork、Common Utility 和 Business Components 组件构成。Common Utility 提供非现场监测业务中涉及的公用服务;Business Components 提供对非现场监测各项具体业务的处理功能,需要使用 Common Utility Layer 提供的服务。

Framework 提供公用的、与业务逻辑无关的服务,它由多个可重用的组件构成。根据非现场监测系统的业务特点和应用需求,Framework 满足以下需求:

（1）请求处理：为应用程序处理所有的 HTTP 请求，HTTP 请求可以是任何形式；

（2）模块化：软件系统由许多作为服务的子系统组装而成。模块化通过局部 的设计和实现的改变使整个系统的性

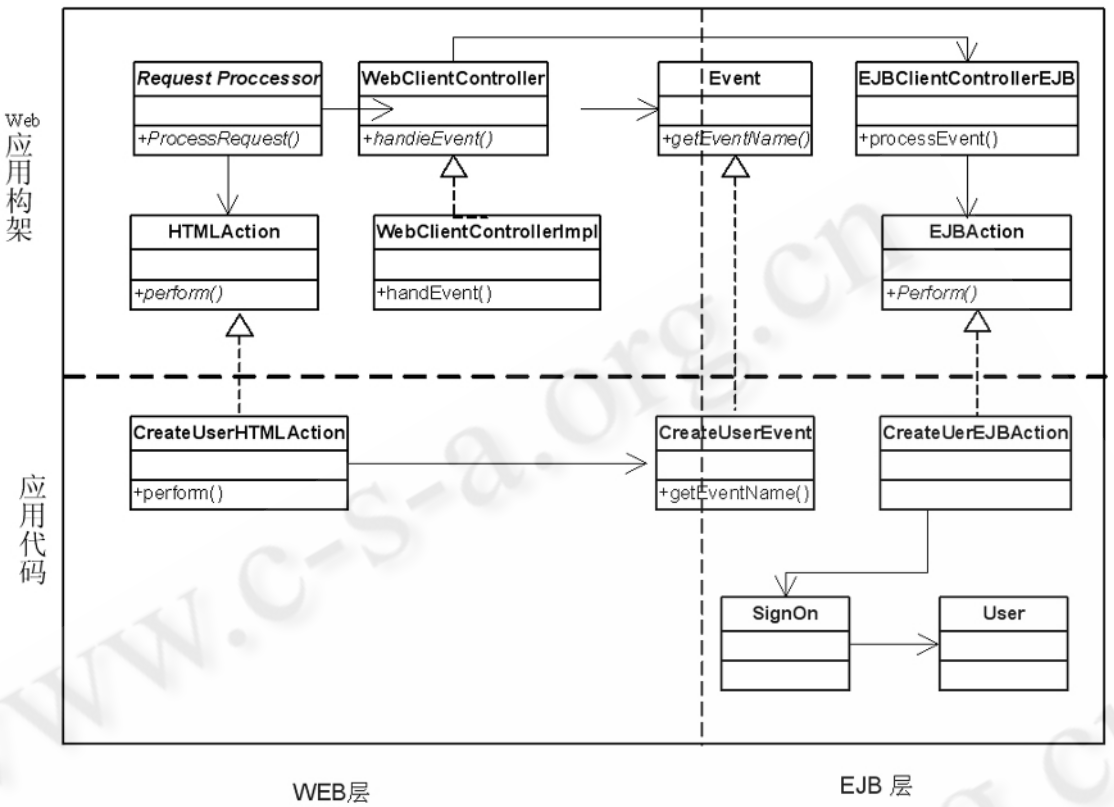

图 2 用户管理构架图

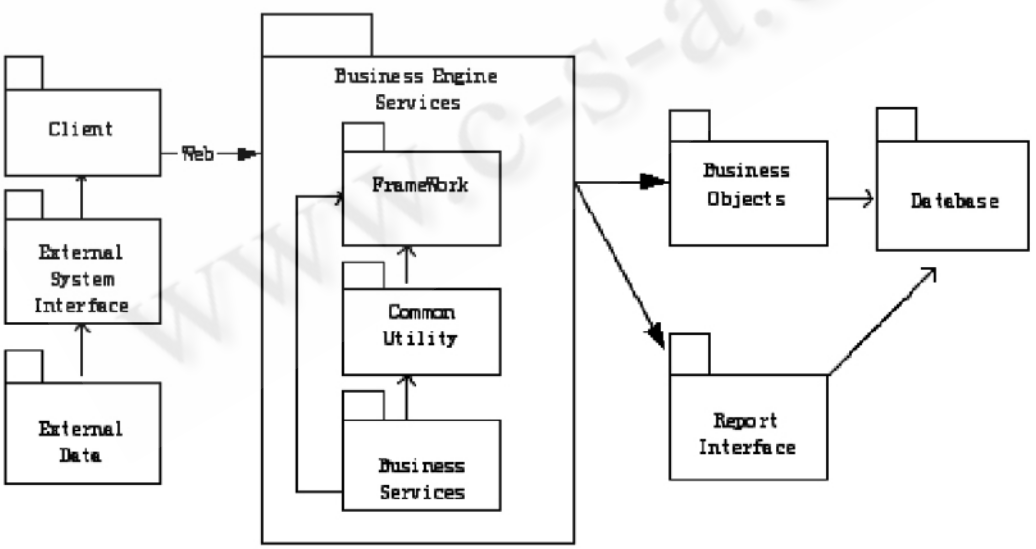

图 3 组件结构图

能得到提升;

(3) 重用性:Framework 中的组件可以被容易的应用于各种项目中;

(4) 可扩展性:允许在运行系统中添加新服务。开发人员能够很容易的添加新的服务。有一个好的扩展机制;

(5) 可维护性:提供系统监测,可以对服务和与服务相关联的资源进行控制和调整。这也包括灵活的记录日志和事件;

(6) 标准性:与 J2EE 组件相兼容。包括 EJB、Servlets、JMS Listeners 等;

(7) 安全性:提供用户验证和资源安全性的控制。

Framework 主要由以下几个组件构成:

MainServlet——负责控制对用户的 HTTP 请求进行处理;

RequestHandling——实现对用户请求的管理。RequestHandling 负责接收用户的请求,将信息进一步传递出去,同时负责响应信息的收集并返回用户(其中 RequestProcessor 负责处理客户端发出的 HttpServletRequest 请求,并通过调用 ActionFormImpl 组件将 HttpServletRequest 中的参数封装在 ActionForm 类中)。

ControllerWebImpl——负责根据封装用户请求的 ActionForm 类选择使用哪个 Controller Session Bean,并将请求传递给选中的 Controller Session Bean 进行处理。

ServiceHandling——实现对不同组件、不同服务的管理。每一个 Service 都有一个 Service name,当用户需要使用 Service 时,Service Management 会根据用户提交的 Service name 调用相应的 Service。

ScreenFlowManager——负责 Web 浏览器页面内容管理。它负责读取 Screen Flow XML 文件,以及根据客户端请求获取显示结果的 JSP 页面的 url 地址。

Persistence——主要定义建立 Entity Bean(实体 Bean)的规范,通过给出建立 Entity Bean 的例子来展现如何建立 Entity Bean。同时定义 session bean(会话 Bean)或普通的 Java bean 通过 JDBC 连接数据库,并给出连接 JDBC 的例子。

ErrorHandling——对系统抛出的异常进行管理。

Security——主要实现系统安全的管理,包括系统登录和退出。

Business Engine 的实现采用基于 J2EE 的构架。J2EE 提供了一个企业级的计算模型和运行环境用于开发和部署多层体系结构的应用,大大简化了分布式应用的开发和复用过程。

5 结束语

与传统的软件开发方式相比,使用复用技术可以减少软件开发活动中大量的重复性工作。组件技术的提出和应用,真正实现了软件复用的思想,从而提高了软件的生产率,减低了开发成本,缩短了开发周期,其优势表现为:

(1) 提高生产率。软件复用最明显的好处在于提高生产率,从而减少开发代价。用可复用的组件构造系统还可以提高系统的性能和可靠性,因为可复用组件经过了高度优化,并且在实践中经受过检验。

(2) 改善软件质量。由于软组件大都经过严格的质量认证,并在实际运行环境中得到检验,因此,重用组件有助于改善软件质量,同时软件中需要维护的部分也减少了。

(3) 提高系统的灵活性。大量使用组件,软件的灵活性和标准化程度也能得到提高。通过使用接口的同一个实现,系统将更为有效地实现与其他系统之间的互操作。

(4) 支持快速原型。即可以快速构造出系统可操作的模型,以获得用户对系统功能的反馈。利用可复用组件库可以快速有效地构造出应用程序的原型。

建立一个应用系统需要重用很多已有的组件模块,这些组件可能是在不同的时间、由不同的人员开发的,并有各种不同的用途。在这种情况下,应用系统的开发过程就变成对组件接口、组件上下文以及框架环境一致性的逐渐探索过程。例如,在 J2EE 平台上,用 EJB 框架开发应用系统,主要工作是将应用逻辑,按 session Bean、entity Bean 设计开发,并利用 JTS 事务处理的服务实现应用系统。其主要难点是事务划分、组件的部署与开发环境配置。概括地说,传统的软件开发过程是串行瀑布式、流水线的过程,而基于组件的开发过程是并发进化式,不断升级完善的过程。

参考文献

1 Michael Girdley《J2EE 应用与 WebLogic Server》,电子工业出版社, 2001。

2 Paul J. Perrone《J2EE 构建企业系统》,清华大学出版社, 2001。

3 Erich Gamma, Richard Helm《Design Patterns Elements of Reusable Object – Oriented Software》Addison – Wesley Longman, Inc. 2000.