基于复用的构件开发模型的研究与应用

亓慧艳[1]，程建平[2]

（1. 山东师范大学信息科学与工程学院，山东 济南 250014；
2. 中创软件工程股份有限公司，山东 济南 250014）

摘 要：提出了一种基于复用的构件开发模型，该模型解决了构件内部结构和组织问题，保证良好的功能职责划分和关注点分离；保证构件以规范化的方式提供对外服务接口和扩展接口；保证构件具有良好的扩展性以及随需应变的能力。通过应用该模型开发了面向金融领域的客户管理构件，并将该构件复用于具体的金融项目。实践表明，该模型能提高软件复用率，降低开发难度，加快开发速度。

关键词：软件工程；软件复用；构件；构件开发模型；基于构件的软件开发

中图分类号：TP311.5 **文献标识码**：A **文章编号**：1674-7720(2010)14-0007-04

Research and application of reuse-based component development model

QI Hui Yan[1], CHENG Jian Ping[2]

(1. Department of Information Science and Engineering, Shandong Normal University, Jinan 250014, China;
2. CVIC Software Engineering Co., Ltd., Jinan 250014, China)

Abstract: This paper presents a reuse-based component development model, which can resolve questions of internal structure and organizational, guarantee good functional division of responsibilities and separation of concerns; guarantee component with standardized way to the provision of external services interfaces and expansion interfaces; guarantee that component has good scalability, as well as on-demand capabilities. Customer management component which applied this model is developed for the financial domain; and then re-used for specific financial items. Practice has indicated that this model can improve the rate of software reuse, reduce the development difficulty and speed up the development pace.

Key words: software engineering; software reuse; component; component development model; component-based software development

随着信息技术的飞速发展，各个行业对软件的需求也迅速增长，软件产品的规模不断扩大，复杂性也不断提高，但目前软件的开发与生产能力却相对不足，形成脱节现象，导致软件危机。软件复用是在软件开发中避免重复劳动的解决方案，被视为解决软件危机、提高软件生产率和质量的有效途径[1]。

软件复用技术是软件工程领域的一个研究热点。复用概念的第一次引入是在 1968 年 NATO 软件工程会议上，Mcllroy 的论文“大量生产的软件构件”中[2]。近十几年来，随着面向对象技术和分布式对象技术的出现并逐步成为主流技术，为软件复用提供了基本的技术支持。构件模型是对构件本质特征的抽象描述，被视为实现成功复用的关键因素之一。构件是核心和基础，复用是必需的手段[3]。尽管软件复用的概念自提出以来已取得了一定的成果，然而在实际软件生产中，真正成功的软件复用项目并不多见。主要原因是受可复用构件的分析和设计方法的限制，针对此问题本文提出了一种基于复用的构件开发模型，即(表现层、业务逻辑层、持久化层)的分层设计模型。

本文中提出的构件开发模型规范和约束构件的结构，保证了良好的功能职责划分，保证构件以规范化的方式提供对外服务接口和扩展接口。对于复杂系统，需求变化不稳定，为使设计具有更好的可扩展性、灵活性与逻辑性，通过分层分而治之，使其关注点分离。在设计中使用模块增进了内聚性，降低了耦合度，进而降低了复杂性并增强了可维护性。通过应用该模型开发了面向金融领域的客户管理构件，并将该构件复用于具体的金融项目。最后，通过将传统的软件开发方法与基于复用的构件开发模型方法进行比较。

1 软件复用与软件构件

1.1 软件复用的基本概念

软件复用是指重复使用“为了复用目的而设计的软件”的过程[4]。软件复用是在软件开发中避免重复劳动的解决方案，它包括对软件生产过程中其他劳动成果的复用，如需求分析、概要设计、详细设计、编码、测试用例和使用手册等。

1.2 实现软件复用的关键因素

实现软件复用的关键因素如图1所示，主要包括：软件构件技术、领域工程、软件构架、软件再工程、开放系统、软件过程、CASE技术等，以及各种非技术因素[5]。本文主要研究软件构件技术对实现软件复用的影响。

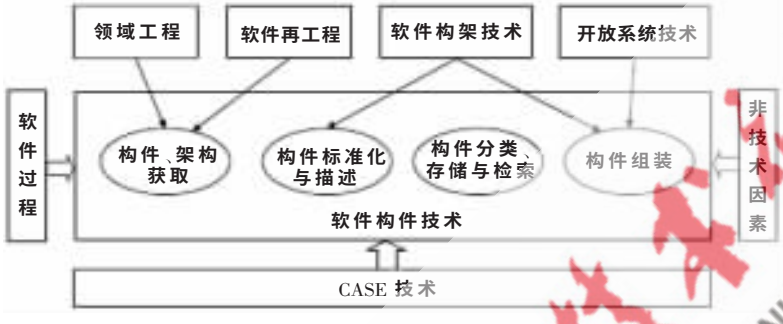

图1 实现软件复用的关键因素

1.3 构件的基本概念

在众多的软件复用开发方法中，基于构件的软件开发方法是一条有效、实际的软件复用途径，所谓构件是指系统中可以明确辨识的构成成分，软件构件是系统中具有一定意义的独立构成成份[6]。

构件应具备的基本特征：(1)复用：复用是构件最基本的性质，构件的设计必须满足能在新的应用项目中使用；(2)封装：封装对外界隐藏构件的设计和实现细节，仅通过接口与外界交互，可以保证构件功能复用的完整性和构件开发及交付的独立性；(3)组装：构件可以通过组装形成新的构件或系统，组装是构件复用的手段；(4)粒度：构件是有大小的，与领域相关的构件粒度大；(5)层次：构件可以按层次进行划分，企业级应用系统的复杂逻辑可以通过分层来解决。

1.4 可复用构件开发遵循的基本原则

(1)“开-闭”原则：一个软件实体应该对扩展开放，对修改关闭。所谓“开”，就是要求构件能适应不同的应用环境；所谓“闭”，就是要求构件是一个独立的整体。这是可复用构件开发最基本的原则，也是复用要求的直接体现。实现“开-闭”原则的基本策略是对变化性进行封装，即找到构件的可变因素，并将其封装起来。

(2)依赖倒置原则：要针对接口编程，不要针对实现编程。它是指导开发可复用框架的基本原则，使得框架无需依赖具体构件，具体构件的开发需遵循框架设定的约束。实现依赖倒置原则的基本策略是在框架中为具体构件设立抽象表示。

(3)接口隔离原则：是一个指导可复用构件实现接口设计的原则，它使得构件中每一个实体都拥有尽可能简单的接口。实现接口隔离原则的基本策略是对构件中的每个角色进行划分，然后针对每一类使用者提供一套有针对性的接口。

(4)迪米特法则：每个软件实体应当尽可能少地与其他实体发生相互作用[7]。实现迪米特法则的基本策略是控制软件实体间由于信息传递而产生的耦合关系。

2 构件开发模型

构件模型是对构件本质特性的抽象描述[8]。本文提出一种构件分层设计模型，如图2所示，该模型解决了构件内部结构和组织问题，保证良好的功能职责划分，保证构件以规范化的方式提供对外服务接口和扩展接口，保证构件具有良好的扩展性以及随需应变的能力。该模型遵循本文中上面所提出的可复用构件开发的基本原则，在设计中使用模块，增进了内聚性，降低了耦合度，低耦合降低了复杂性并增强了可维护性。对于复杂系统，需求变化不稳定，为使设计具有更好的可扩展性、灵活性与逻辑性，通过分层分而治之，主要分为表现层、业务逻辑层、持久化层，使其关注点分离。API(Application Programming Interface)接口是外部调用者；MAI(Management Application Interface)接口是内部调用者；SPI(Service Provider Interface)接口是构件本身的可扩展点，通过分层带来更好的可伸缩性与可维护性。其中，表现层又叫展现层，主要功能是向人展示，最终实现人机交互。一般表现层与具体的业务需求有关，需具体问题具体分析，在此就不再详细介绍，下面主要介绍业务逻辑层设计模型和持久化层设计模型。

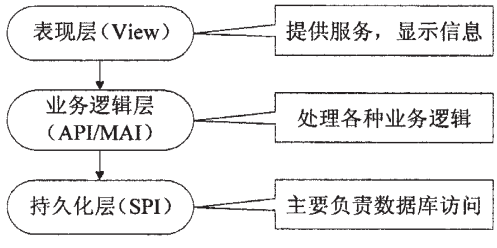

图2 构件分层设计模型

2.1 业务逻辑层设计模型

业务逻辑层设计模型如图3所示。业务逻辑层根据职责具体分为服务层和领域层，(1)Service服务层定义了应用的边界和从客户角度所能看到的可用操作集。①它包括实体home即Entity Home和流程管理，由Normal Service实现；②它是无状态类，不包含任何业务逻辑，定义了一个公共的应用操作集合，可供各种使用者使用。(2)领域层的主要职责是实现业务逻辑，它包含API层和

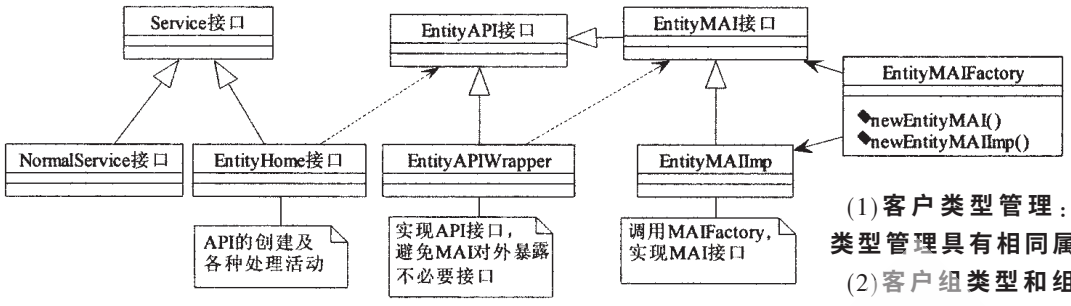

图 3 业务逻辑层设计模型

MAI 层。①API 层遵循最小化设计原则，在完成功能的前提下尽可能简单，提供相对稳定的接口，减少改动带来的影响，易于使用；EntityAPIWrapper：实现 API 接口，避免对外暴露 MAI 接口，实现延迟加载等；②MAI 层是内部调用者，EntityMAI：定义可以对外提供的服务和职责；EntityMAIBase：提供基本共性实现；EntityMAIImp：为内部调用者提供服务，调用 SPI 层实现持久化。

2.2 持久化层设计模型

持久化层设计模型，如图 4 所示，持久化层主要功能是实现数据的存储、让客户了解功能的扩展点和为业务逻辑层提供服务。(1)SPI 接口：定义抽象共性的方法；(2)Base 实现类：定义共性的方法，对对象进行延迟加载；(3)JDBC 实现类：负责与数据库交互；降低对底端的依赖，增强变化性；(4)Wrapper 实现类：负责创建对象时对返回的对象进行包装，对日志、缓存和关系逻辑进行处理，在批处理过程中可以结合 cache 及 Pool 进行预加载。

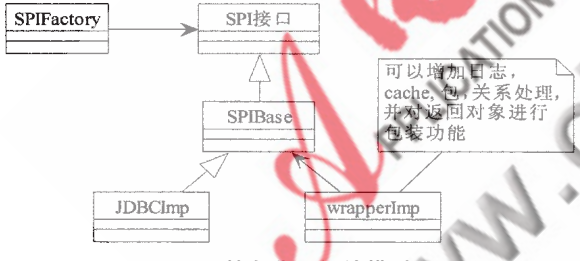

图 4 持久化层设计模型

3 基于构件的软件复用开发实例

结合本人在中创软件公司实习中参与的项目实践，本文应用该模型开发了面向金融领域的客户管理构件，保证构件以规范化的方式提供对外服务接口和扩展接口，保证构件具有良好的扩展性以及随需应变的能力，并将该构件复用于具体的金融项目，降低了开发难度，加快了开发速度，能够较好地解决复用这一问题，能够从构件开发模型结构的高度指导构件的开发和灵活复用。

3.1 客户管理构件

客户管理构件是用于对客户信息与客户间相互关系进行管理维护的构件，提供统一的客户管理功能，提供可视化的客户信息定义，满足个性化需求。不同行业、不同的企业对客户信息的管理内容千变万化，客户管理构件是面向领域的构件，本文中所研究的是面向金融领域。客户管理构件总体上分为以下几个模块：

(1)客户类型管理：客户类型可配置，通过类型管理具有相同属性的客户集合；

(2)客户组类型和组管理：通过组来管理具有不同特征的客户集合；

(3)客户信息维护：通过客户类型，对客户信息的增加、删除、修改、查询和版本控制等进行统一维护；

(4)客户关系维护：管理不同类型客户之间的相互关系。

3.2 金融租赁客户管理系统

利用客户管理构件，针对金融租赁客户管理系统项目具体需求，支持该项目客户管理系统的开发。金融租赁客户管理系统包含的主要子领域及其相互之间的依赖关系如图 5 所示，各个子领域的简要说明如下：

(1)客户信息查询与维护：该领域封装了通过客户类型对客户信息表中记录的查询、增加、删除、修改及维护功能；

(2)个人客户管理：主要负责对金融租赁领域中涉及的自然人的信息进行管理；

(3)对公客户管理：主要负责对金融租赁领域中涉及的组织机构进行管理；

(4)集团客户管理：主要负责对两个或两个以上客户成员团体的信息进行管理；

(5)同业客户管理：主要负责对金融机构客户的信息进行管理；

(6)部门及管护员调整：主要负责对客户部门、经理发生变动的信息进行管理；

(7)客户信息数据库：主要负责对整个客户管理信息

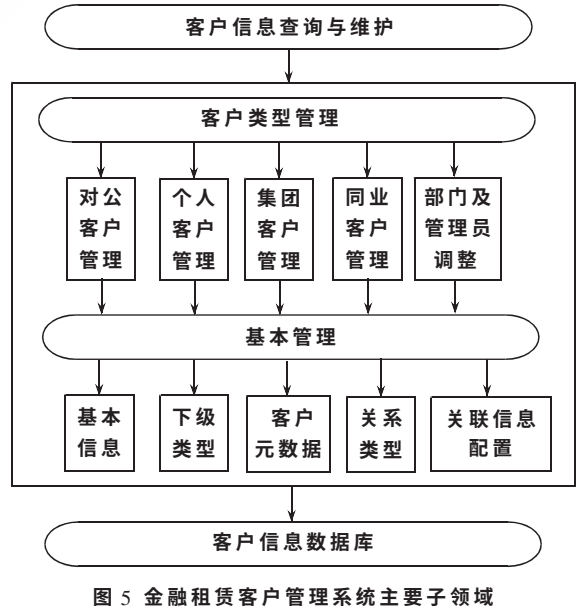

图 5 金融租赁客户管理系统主要子领域

领域中涉及的各种客户信息以及其他相关信息进行存储并提供访问接口。

在金融租赁客户管理系统中运用了大量的构件，主要有：用户权限管理构件、参数管理构件、元数据管理构件、财务报表管理构件、评级管理构件、内容管理构件等。构件代码比例约占62.3%，新增构件约占28.6%，构件化比例相当大，使原本半年的开发时间缩短为两个月，减少了开发成本，提高了软件开发效率和产品的质量，减少了系统的维护代价。为了更好地说明此构件开发模型的优势，将传统的软件开发方法与基于复用的构件开发模型方法进行比较，统计结果如表1所示，实践表明，基于本文提出的模型开发的构件降低了开发难度，加快了开发速度，较好地解决了复用的问题。

表1 传统的软件开发与基于复用的构件开发模型方法对比

软件开发方法 / 名称	传统的软件开发	基于复用的构件开发模型
代码复用率/%	25.7	72.3
开发周期/月	6	2
开发成本/%	100	原成本的60%，降低约40%
可扩展性	可扩展性差，约为30%	可扩展性达80%，直接复用已有构件或做适当修改
可移植性	环境独立性低	独立性好，可运行在不同平台和软件环境中
可修改性	模块性差，通用性低	模块化程度高，通用性好
可维护性	可维护性低，管理不方便	通过版本控制管理，可维护性高，易于管理
灵活性	灵活度低	灵活度高，自动化程度高

4 总结与展望

本文提出一种基于复用的构件开发模型，该模型规范和约束了构件的内部结构，保证构件以规范化的方式提供对外服务接口和扩展接口；保证构件具有良好的扩展性以及随需应变的能力；将该开发模型应用于客户管理构件开发，并基于该构件复用于金融租赁客户管理系统。通过实践对比表明，使用该构件开发模型降低了开发难度，加快了开发速度，能够较好地解决复用这一问题。由于构件开发涉及领域繁多，该构件开发模型还有很多不足之处，还有待于在实践中继续完善。

参考文献

[1] 杨芙清，梅宏，李克勤.软件复用与软件构件技术[J].电子学报，1999，27(2)：68-75.

[2] MILI H. IEEE transactions on software Engineering, June 1995, 21(6):528-562.

[3] MCCLURE C. Software reuse: a standards-based guide [C].Wiley-IEEE Computer Society, 2001.

[4] WALLNAU B. The current state of CBSE[J]. IEEE Software, 1998,15(5):37-46.

[5] PRESSMAN R S. Software engineering: a practitioner's approach [M]. 北京：机械工业出版社，2005.

[6] 杨芙清，梅 宏. 面向复用的需求建模[M].北京：清华大学出版社，2008.

[7] 阎宏. Java与模式[M].北京：电子工业出版社，2002.

[8] EVANS E. Domain-driven design: tackling complexity in the Heart of Software[M]. Addison Wesley, 2006.

(收稿日期：2010-01-29)

作者简介：

亓慧艳，女，1986年生，硕士研究生，主要研究方向：软件工程。

程建平，女，1954年生，研究员，硕士研究生导师，主要研究方向：计算机软件、软件工程。