文章编号：1005-8451 (2010) 09-0051-04

基于领域驱动设计的软件开发方法和实例分析

陈 亮

(广州铁路（集团）公司 科学技术研究所，广州 510100)

摘 要：研究了企业级应用系统开发的现状，明确采用基于Web的多层架构体系(如ASP.NET)来进行企业级应用开发，分析数据库驱动设计方法在Web应用开发中存在的缺点，引入领域驱动设计方法，介绍了领域驱动设计方法的开发模式，并结合动态表单的设计中实例，完成系统的分层架构和领域建模，解决基于数据库驱动设计方法的Web应用开发存在的诸多问题，使系统获得很好的扩展性和可维护性。

关键词：领域模型；数据库驱动设计；领域驱动设计；软件开发方法

中图分类号：TP39 **文献标识码**：A

Developing method of software based on design of domain driven

CHEN Liang

(Science and Technology Institute, Guangzhou Railway (Group) Corp, Guangzhou 510100, China)

Abstract: The current situation of the Enterprise Application Systems development was researched, using a Web-based multi-storey structure architecture (Asp.Net) for enterprise-class application development was cleared, the shortcomings of database-driven design method in webbased application development were analyzed, omain-driven design method and its development model were introduced and applied to the Dynamic Table System, the layered architecture and domain modeling of SPP System were completed,many problems of Web application development based on database-driven design method were solved and made the System expaysibility and maintainability.

Key words: domain model; database-driven design; domain-driven design; method of software development

当今，Web应用的重要性已经得到了广泛认同，越来越多的应用系统都被构建在Web之上。基于Web的多层架构体系（如ASP.NET）已经成为解决企业级应用的主要途径。然而，传统的数据库驱动设计方法（以数据库建模为核心的软件开发方法）在基于Web多层架构的企业级应用系统开发中存在诸多不足。本文在分析了数据库驱动设计方法存在不足的基础上，引入领域驱动设计（DDD）方法，并研究了这种设计方法在动态表单系统开发中的应用。

收稿日期：2010-01-25

作者简介：陈 亮，工程师。

1 数据库驱动设计方法存在的问题

1.1 不能有效地反映需求

面向对象的思维方式符合人类的自然思维，因此采用OOA（面向对象分析方法）进行需求分

数据来更新线路单元实体，然后分别从屏幕刷新它们，连接单元的工作便完成了。

4 结束语

使用单元法来进行铁路线路的平面设计，较之传统的交点法布线手段更加灵活，修改方便，适应性更强，可以提高的工作效率。而通过利用自定义对象，直接将数据存储于图形文件中的方法，使应用程序不必访问外部数据库便能轻松实现数据的存取，让操作更加直接、方便和快捷。由于所有数据均存储于单一的图形文件中，对数据的维护和管理也更加的轻松简单并且更不易出错。

参考文献：

[1] 邵俊昌，李旭东. AutoCAD ObjectARX 2000开发技术指南[M]. 北京：电子工业出版社，1999.

[2] 李世国，潘建忠，平雪良. AutoCAD 2000 ObjectARX编程指南[M]. 北京：机械工业出版社，2000.

[3] 詹振炎. 铁路选线设计方法的现代理论和方法[M]. 北京：中国铁道出版社，2001.

[4] 郝 瀛. 铁道工程[M]. 北京：中国铁道出版社，2007.

析可以使得软件的需求（问题域）和实现（解决域）在形式上是一致的，很便于理解，因此采用面向对象数据库是首选。然而，对于OODB（面向对象数据库）来说，从技术上还不够成熟，商品化的OODBMS尚不普及，目前主流的DBMS（Microsoft SQL Server，Oracle，DB2等）都是关系型的。

传统的以数据库建模为核心的软件开发是基于关系型数据库的。这样必将导致需求分析(问题域)与系统实现（解决域）在形式上不一致，影响设计人员的设计思路，甚至使整个系统的设计偏离用户需求。现代的软件系统构架应该是问题域建模和模式，也就是说，在做系统设计的时候，首先要搞清楚对象：当前的问题是什么？解决这个问题需要什么样的逻辑和业务流程？选择什么样的模式或软件框架才可以解决这样的问题，并使得软件系统具有鲜活的生命力？在现代的软件构架模式中，数据库的重要性已经显得不太明显了，它只不过是一种数据持久化的机制，与问题域建模一点关系都没有。

1.2 导致过程化编程

现在，面向对象程序设计已被人们广为接受，传统的面向过程程序设计方法在系统的可维护性、可扩展性方面存在严重瓶颈。而以数据库建模为核心的软件开发方法容易导致过程化程序设计。因为数据库建模，首先是根据需求确定数据库结构，然后程序员编写过程化的SQL语句、存储过程。这些过程化元素将导致业务逻辑无法用完全面向对象（OO）的思想来实现，从而导致开发出来的系统可维护性和可扩展性低下。

软件系统的设计要求能够考虑到由于问题域矛盾的变化而带来的功能性甚至结构性的变化。在首先确定数据库设计的软件设计中，在更改业务逻辑层的处理过程之前还需要首先考虑数据库设计的变更。表面上看，软件3层体系结构体现得很完善，各层分工很明确，其实这种做法仍然没有降低业务逻辑层和数据持久层的耦合度。

1.3 运行性能影响

围绕数据库分析设计容易导致软件运行时负载集中在数据库端，系统性能难于扩展（走上集中式、昂贵的、高风险的大型机模式），闲置了中间件（如.Net服务器）分布式集群处理能力，就是使用了集群，也分担不了负载。

软件系统设计应该把重点放在问题域建模以及模式和框架的选择上。问题域建模可以解决实际的业务逻辑问题，模式和框架则是软件生命力的重要保障，数据在问题域建模中流动，数据库是数据流动的终点。从理论上来讲，数据完全可以持久化到文件，或者只要能够保证足够的内存以及服务器永不停机，数据完全可以高效地保存在内存中，这样也就没有使用数据库的必要了。当然也不是数据设计不重要。应该把系统设计重点放在问题域上，而不是数据库上。数据库仍然是现在主流的数据持久化机制，但只负责数据持久化，它并不能决定业务流程。事实上，现在的Hibernate和NHibernate就是一个数据持久化框架，它能够使得软件设计和开发人员摆脱数据持久层的实现细节，将业务逻辑层和数据持久层的耦合度降到最低。

2 软件的领域和模型

软件总会与其用户的活动和兴趣相关，用户在使用软件中主要的环境称为软件的领域。软件开发的目的就是通过计算机解决某一领域的实际问题。这样的定义已经将立足点置于领域层面了：需要关注的是领域本身，而不是其它的技术细节。模型是知识的一种有选择的简化和有意识的组织形式，一个合适的模型可以有效地表达软件领域，目前流行的软件模型表示方法为UML。

2.1 领域驱动设计与模型驱动设计

领域驱动设计应该是建立在模型驱动设计的基础上，但两者有所不同，主要表现在领域模型不仅仅是模型，还包括与模型相关的用来完成与业务逻辑相关的其他元素，比如服务等。此外领域模型还具有独有的特性：完成性约束，相互关联和访问机制。

2.2 领域的组成

通常将领域中的组成角色分为以下5种：实体(Entity)、值对象(Value Object)、工厂(Factory)、仓储（Repository）、服务（Service）。

（1）实体：实体是在做开发的时候经常遇见的领域对象。实体是一种领域对象，在特定的上下文（Context）中，需要关心的不仅仅是“它是什么”，要更深入地知道“它是哪个”，因此，实体需

要有自己的唯一标识符。以标识作为基本定义的对象称为“实体”。

（2）值对象：如果一个对象代表了领域的某种描述特征，并没有概念性的标识，那么它就是值对象。

（3）工厂：定义创建实体的方法。它的职责，就是创建对象。既然是创建对象，那我们为什么不直接实例化呢？简单的情况是可以的，往往会给工厂带来巨大的好处，简单地说就是屏蔽了创建对象的复杂性。

（4）仓储：仓储的概念和一些人常说的数据访问对象（DAO）有些类似，但是并不等同，二者一个很大的不同是仓储有“根”的概念，而数据访问对象往往是按照数据库的表来划分的。

（5）服务：服务不属于实体也不属于值对象，代表的一些重要领域操作，是一些本质的行为和动作，被执行的操作涉及到领域中的其他的对象，除了操作外，服务是无状态的。当领域中的一个重要的过程或者变化不属于一个实体或者值对象的自然职责时，向模型中增加一个操作，作为一个单独的接口将其申明一个服务。

3　领域驱动设计的系统分层架构

Web系统普遍采用MVC的分层架构，在基于领域驱动设计的系统中，层次进一步细化，领域模型得到隔离，见图1。

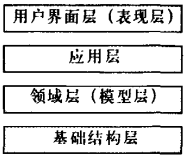

图1　系统分层架构图

（1）用户界面层（表现层）：这一层封装了服务于访问系统的客户端所需的所有表现逻辑，拦截了客户端的请求，提供了单一的登陆入口，构造了会话的管理。控制了对业务服务的访问，构造了响应，并且把这些响应传递给客户端。

（2）应用层：定义了系统要完成的工作，不负责业务逻辑的实现，而是指导下层的领域层来进行业务逻辑的实现工作。

（3）领域层（模型层）：负责系统业务逻辑的实现工作，这一层是业务软件的核心。

（4）基础结构层：为上层提供通用的技术能力：应用的消息发送、领域持久化，为用户界面绘制窗口等。该层还负责与外部资源、外部系统（比如数据存储和遗留应用系统）通信。

4　实例分析

动态表单系统是设计人员根据实际需求，设计好表单格式，然后用户根据自己的实际情况填写表格所要求的内容。本文以动态表单的分析作为实例，基于DDD开发模式为指导，通过和领域专家的多次交流，迭代得到一个动态表单的领域模型，如图2。

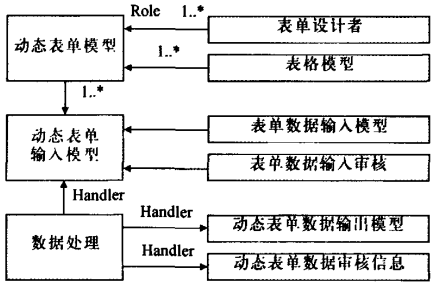

图2　动态表单领域模型

4.1　精简关联关系

用户（Designer）和动态表单（DynamicTableInfo）是一个多对多的关系，为了减少复杂性，指定一个导向，指定这种关联是从DynamicTableInfo到Designer的，也就是说，所关心的是DynamicTableInfo是由哪个Designer来操作，这样子多对多的关系就可以变成一对多的关系，再通过Role限定Designer的不同身份，进一步将一对多的关联精简成一对一的关联。

4.2　区分实体和值对象

TableModel不具有连续性标识，只是描述DynamicTableInfo的一个值对象；Designer的标识（DesignerID）必须在整个动态表单系统中进行跟踪，因此Designer是实体；DynamicTableInfo是动态表单的必备数据，必须有连续的标识，因此是实体，DynamicTableOutPut和DynamicTableCKInfo是动态表单的实际输出数据和审核数据，具有连续的标识，也是实体。

4.3　服务

DealHandler不是实体也不是值对象，这个只是负责为用户提供处理DynamicTableInfo的服务。因此它是服务。

4.4 领域对象的管理

4.4.1 聚合边界的设置

出于数据库多用户访问时的安全问题，需要为一簇相关联的对象设置边界，以控制用户的访问方式。DynamicTableOutPut，DynamicTableCKInfo，DynamicTableInput，都有自己的标识，并且有时需要单独对其进行跟踪。因此其是自己的聚合根。TableModel，Designer，DynamicTableInfo应该放入一个聚合体内，因为无需单独对TableModel，Designer进行跟踪。DynamicTableInfo是这个聚合体的聚合根。

4.4.2 选择仓储

只有作为聚合根的实体才能够拥有仓储。本系统中聚合根有DynamicTableInfo (聚合体的聚合根)、DynamicTableOutPut，DynamicTableCKInfo，DynamicTableInput，相应的仓储为DynamicTableInfo Repository、DynamicTableOutPut Repository、DynamicTableCKInfo Repository、DynamicTableInput Repository。仓储在实现阶段可以采用DAO模式，图3展示了部分模型的聚合边界设置和聚合根的仓储定义。

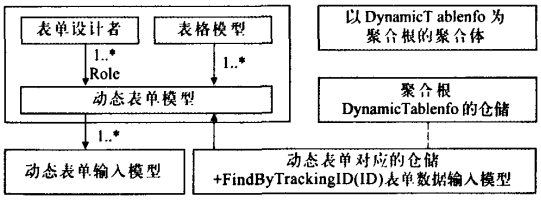

图3 部分模型聚合根的仓储定义

4.4.3 对象创建

基于动态表单领域对象的复杂性，为此采用工厂模式来进行对象的创建，将复杂的创建任务从领域对象中分离到工厂对象中。根据不同的情况可采用相应的设计模式实现工厂模式：工厂方法，抽象工厂和构造器。如图4为DynamicTableInfo对象的工厂构造过程。

4.5 编码实现

根据领域模型编码实现系统功能。

4.6 重构

对软件进行重新设计而不改变软件的功能，

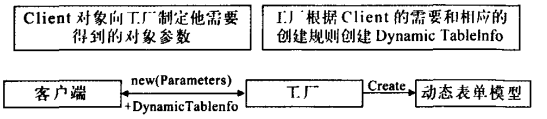

图4 对象的工厂构造过程

重构不要求预先做出各种详细设计，而是对代码进行一系列小的独立的修改，每次修改都保持功能不变，使得设计更加灵活易懂。模型重构是基于对领域的理解来执行的，通常包含一系列的微重构，逐渐产生一个更加合理的模型。每次获得新的领域知识或者深入的理解之后，都应该将模型重构到更深的层次。

5 领域驱动设计的优点

在利用领域驱动设计（DDD）方法进行系统建模以及后续的开发过程中，系统带来了许多优点。

（1）基于模型驱动开发（MDD）思想的DDD采用面向对象的思维方式，符合人类自然思维。

（2）基于DDD方法构建的上述动态表单模型，能够很自然的引导程序开发人员采用面向对象思路实现系统，从而确保开发出的动态表单系统具有很好的可维护性和可扩展性。

（3）采用DDD开发出的系统能够充分地利用中间件（如.Net服务器）分布式集群处理的能力，减轻了数据库端的负载过重问题。

6 结束语

通过对软件领域的分析，得到一个领域模型，在领域模型的基础上进行软件的设计与实现工作，通过重构的方法进行领域模型的精化工作。

参考文献：

[1] 甄 镝. .Net与设计模式[M]. 北京：电子工业出版社，2006.

[2] Eric Evans. 领域驱动设计[M]. 陈大峰，张泽鑫. 北京：清华大学出版社，2006.

[3] 汤 晨，吴朝晖. 一个利用模型驱动体系结构技术的分布式系统实现[J].计算机工程与应用，2003，(33)：133-135.

[4] WhiteC. xSLT从入门到精通[M]. 王 健，王 军. 北京：机械出版社，2003.