基于领域驱动设计的软件开发

王 忠,程 磊

(武汉工程大学 计算机科学与工程学院,湖北 武汉 430074)

摘 要:领域驱动设计(Domain- Driven Design)是以敏捷开发为手段,以模型驱动设计为根基,以软件领域为着眼点。应用领域驱动设计可以快速应对软件开发中的变化。领域驱动设计核心是建立一个单一的既符合软件所处领域本身又适合软件分析开发需要的领域模型,着重介绍了领域驱动设计的相关概念以及开发流程。

关键词:领域驱动设计;领域模型;重构

中图分类号:TP311.52 文献标识码:A 文章编号:1672- 7800(2008) 02- 0037- 03

大多数情况下,软件的分析与设计是分裂的,分析阶段得到的模型往往在设计实现阶段体现不出应有的价值,主要有3个原因:①没有重视模型的价值,分析模型中提炼的各种概念、知识并没有在设计中得到体现;②模型与具体软件开发语言环境脱节,模型不能正确地应用到具体的软件开发语言环境中;③是分析得到的模型过于繁琐,模型边界不清晰,关系过于复杂。

领域驱动设计(Domain- Driven Design)[1]抛弃了分裂分析模型与设计的做法,使用单一的领域模型来满足分析与设计这两方面的要求,通过这个领域模型将领域专家与技术人员联系在一起。

1 领域驱动设计的前提与原则

1.1 前提

领域驱动设计的前提有两个[1]:

(1)对于多数软件项目,主要的焦点应该在领域及领域逻辑方面。软件的分析设计与实现过程中,应该始终把提炼领域,表达领域以及实现领域放在第一位,而那些与领域关系不大的问题,比如对象的持久化,对象的界面显示等问题应该放在次要位置,至少在分析与设计阶段尤应如此。

(2)复杂的领域设计应该基于一个模型来进行。在领域驱动设计的过程中,需要一个展示复杂领域知识的平台,这个平台就是模型,通过对领域知识的不断获取与升化,模型也逐渐趋向领域本身。

1.2 原则

(1)将领域模型作为领域专家、分析人员、开发人员之间交流沟通的核心。从这个层面上来讲,领域模型也是一种特殊的语言,领域专家、分析人员、开发人员之间相互交流的语言。领域模型为领域专家提供一个规范的领域表达形式,帮助分析人员理解领域知识,帮助开发人员了解他所从事的特定领域,提高建模能力。

(2)领域模型与设计紧密联系。领域模型与设计的脱离会导致领域模型与软件设计之间映射复杂,当设计发生变化时很难通过复杂的映射关系来修改领域模型,反之亦然。

一个设计或者它的核心部分,不能够映射到领域模型上,这个模型是没有价值的,软件的正确性也值得怀疑。

(3)采用迭代方法提炼领域模型。有价值的领域模型通常不会立刻产生,需要对领域进行深入的理解。每次增加对领域的理解后,领域模型将被改变以反映进一步获得的领域知识,设计与实现也随之改变来反映更深层次的领域模型。

2 软件的领域与模型

软件总会与其用户的活动与兴趣相关,用户在使用软件中主要的环境称为软件的领域。建立与用户活动有价值的软件,开发人员必须把重点放在软件的领域中;模型是知识的一种有选择的简化和有意识的组织形式,一个合适的模型能了解信息的含义并聚焦于问题本身,应用模型可以有效地表达软件领域,目前最为流行的软件模型表示方法为UML[2]。

领域模型的表达方式多种多样,标准情况下用UML图但并不局限于UML图,领域模型可以配备描述(需求采集中的口头描述,或文档中的文字描述),将图形所代表的意义,以及图形中没有呈现出来的规则、断言、细节进行补充,完整地展现软件的领域。

2.1 领域驱动设计与模型驱动设计

领域模型是建立在模型驱动设计中模型的基础上,但是两者又有所不同,主要表现为领域模型不仅仅是模型,还包括与

作者简介:王忠(1981-),男,湖北汉川人,武汉工程大学计算机学院副教授、硕士生导师,研究方向为信息隐藏与数字水印技术;程磊(1968-),男,湖北江陵人,武汉工程大学硕士研究生,研究方向为软件工程、搜索引擎技术。

模型相关的用来完成与业务逻辑相关的其它元素，如服务等，此外领域模型具有独有的特性：

(1) 领域模型具有完整性约束。完整性来源于软件领域本身，现实世界中很多事物都是由多个部分构成，与现实相对应的软件必然也包含了丰富的完整性约束关系。比如一辆轿车必须包括四个轮胎，这个就是轿车的完整性约束，在领域模型中需要提供对完整性约束的维护，保证领域对象的完整性。

(2) 领域模型相互关联。面向对象的软件中几乎没有孤立的对象，对象之间或多或少的存在着不同程度的关联，领域模型也不例外。领域驱动设计强调在领域层中维护领域对象的相互关联。将模型中的元素根据软件领域的不同，划分成相对独立的不同模块，模块与模块之间低耦合，模块内部高内聚，每个模块都是一个相对独立的领域聚集，每个模块都有一个对象作为该模块的根元素。

(3) 领域模型具有访问机制：包括领域工厂，领域对象生命周期的维护等等。

总之，是否突出业务领域是领域驱动设计与模型驱动设计的一个重要标志[3]，领域驱动设计时刻以软件的业务领域如何在软件代码中有效的体现为着眼点。

2.2 领域模型的分类

2.2.1 实体

现实世界中许多事物不是由属性来定义，而是通过一系列的连续性(continuity)和标识(identity)来定义的，与这些事物对应的领域模型元素对象同样并不主要是由它们的属性来定义，它们体现了标识在时间上的延续性，它们可能要经历许多不同的形态，有时一个对象与其他的对象有着相同的属性。以标识作为基本定义的对象称为“实体”。

2.2.2 值对象

如果一个对象代表了领域的某种描述性特征，并没有概念性的标识，那么它就是值对象。值对象就是那些在设计中只关心它们是什么，而不关心它们谁是谁的对象。值对象一般都是临时对象，在一个操作中创建出来，然后马上就被丢弃了，值对象经常作为实体的属性和其他值。

2.2.3 服务

服务不属于实体也不属于值对象，它们代表一些重要的领域操作，是一些本质的行为和动作，除了它们掌握的操作外，服务没有自己的任何状态，在领域中也没有什么意义。但是，如果把这些领域需要的行为和动作交付给实体或者值对象的时候，就会破坏模型中对象的定义，人为添加了一些没有意义的对象。当领域中一个重要的进程或转换操作不是实体和值对象本身的职责时，把操作作为一个独立的接口加入模型，并声明为服务。

3 领域驱动设计的系统分层架构

目前，在开发基于web系统中普遍采用MVC的分层架构，把系统分成模型层，视图层和控制层[4]，在基于领域驱动设计的系统中，层次得到进一步的细化，领域模型得到隔离：

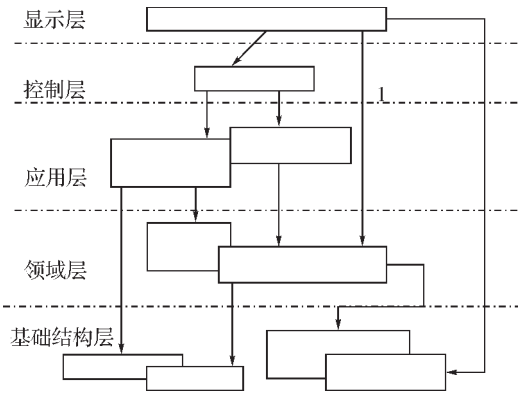

图1 基于领域驱动设计的分层架构

(1) 显示层。负责向用户显示信息，并且解释用户命令。

(2) 控制层。完成业务的流程控制。

(3) 应用层。定义软件功能，调用领域对象来解决问题。应用层这个层保持简练，它不包括处理业务规则或知识，只是给下一层中领域对象协调任务，委托工作。

(4) 领域层。负责表示业务概念，业务状况的信息以及业务规则。反映业务状况的状态在该层中被控制和使用。这一层是业务软件的核心。

(5) 基础结构层。为上层提供通用的技术能力：消息发送，领域持久化，绘制界面窗口等。

4 领域驱动设计的基本流程

领域驱动设计的流程一般为需求获取，模型设计，将模型提炼成领域模型，代码开发，通过重构进一步深入理解领域然后迭代开发。

(1) 需求获取。获取需求过程也是一个迭代的过程，在领域专家的参与下通过多次迭代得到一个轮廓分明的模型，把需求涉及到的所有概念在模型中描述出来，该模型组织了领域知识，并为开发团队提供核心语言。

(2) 区分实体和值对象。按照从简单到复杂的顺序对模型中的对象进行考察，判断对象是属于实体还是值对象，判断的标准参照前文关于领域模型分类讨论。

(3) 约束领域模型之间的关联关系。领域模型之间存在着大量的关联关系，对于模型中的每个可导航(traversable)的关联，在软件中都得提供一种机制来实现与管理关联，特别是双向关联，给实现与维护带来很大的困难，对关联关系进行严格的约束，通过加入限定来有效减少关联的多重性，消除不必要的关联。

(4) 确定聚合的边界。根据领域知识将关联领域对象形成聚合，并确定聚合的根，管理聚合之间的不变量关系。

一个聚合是一簇相关联的对象，每个聚合都有一个根和一个边界，边界定义了聚合中包含什么，根是包含在聚合中的单个特定的实体，是聚合中唯一允许被外部对象引用的元素，通过根来控制所有边界内对象的访问。

在聚合成员之间存在不变量(invariant)关系，(所谓不变量指无论何时数据发生变化都必须满足的一致性规则)由于根控

制了访问,无法饶过根修改聚合内的元素,这就保证了在任何状态变化中,聚合中对象的不变量可以被满足。

(5)选择仓储。对聚合根进行考虑,根据应用需求决定哪些聚合根需要仓储。

仓储提供一套方法将聚合从持久层中提取出来,包括一系列的查询,保存根的方法,以及从根开始访问聚合内其他领域模型的方法。在设计仓储的时候要根据软件领域确定哪些聚合需要提供从持久化层复原的方法,可以通过仓储工厂的形式来组织仓储。

(6)编码实现。根据领域模型编码实现系统功能。

(7)重构。重构[5]是指对软件进行重新设计而不改变软件的功能,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　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] Eric Evans 著.领域驱动设计[M].陈大峰,张泽鑫译.北京:清华大学出版社,2006.
[2] Rumbaugh著.UML参考手册[M].北京:机械工业出版社,2005.
[3] 彭晨阳.领域驱动一切[J].软件世界,2007.
[4] 王一宾,李新科.软件体系结构设计方法的研究[J].计算机工程与设计,2005(3).
[5] Martin Fowler著.重构——改善既有代码的设计[M].侯捷,熊杰译.北京:中国电力出版社,2003.

(责任编辑:余昶颖)

Base on Domain Driven Design Developing Software

WANG Zhong ,CHENG Lei
(School of Computer Science and Technology ,Wuhan Institute of Technology,Wuhan 430074,China)

Abstract:Domain Driven Design (DDD) is a art of design which base on the Agile Development and model driven Design(MDD).The base of the DDD is Software Domain. Using the DDD can quickly respond to the changes in the software development. The core of DDD is how to build a domain model which conforms to the software fields and the needs of the development of the software. This paper introduces the concept and workflow of Domain driven design.

Key Wrods:Domain Driven Design; Domain Model; Refactor