

一种从用例模型到分析模型的设计方法^①

李 军

(长沙环境保护职业技术学院 信息技术系, 长沙 410004)

摘 要: 由于分析模型的复杂性, 软件设计师在软件建模过程中往往容易忽略它, 直接从用例模型到设计模型, 但这样方法难以验证用例的准确性。本文提出了一种从用例模型到分析模型的设计方法, 并阐述了这种方法的机理, 详细说明了从用例模型中识别分析类的方法, 深入探讨了从用例模型中获取实体类的属性及设计分析类的职责。

关键词: 用例模型; 分析模型; 分析类; 统一建模语言; 统一软件开发过程

Method for Designing Analysis Model Based on Use Case Model

LI Jun

(Department of Information Technology, Changsha Environmental Protection College, Changsha 410004, China)

Abstract: Software designers usually overlook Analysis Model Because of its complexity, and make Design Model directly from Use Case Model. But this method cannot verify the accuracy of the use cases. This paper presents a modeling method of firstly from Use Case Model to Analysis Model, then mapping Analysis Model to Design Model, and describes the principle of this method, explains specifically how to distinguish Analysis Class from Use Case Model, and discusses deeply how to obtain the attributes of Entity Class and how to design the operations of Analysis Class.

Key words: use case model; analysis model; analysis class; UML; RUP

1 引言

统一软件开发过程 RUP 中定义了四个模型, 即用例模型 (Use Case Model), 分析模型 (Analysis Model), 设计模型 (Design Model) 和实现模型 (Implementation Model)^[1]。用例模型是其他三个模型的基础, 它是用来规定系统开发需求, 代表了实际业务转化为计算机功能性需求以后的结果。分析模型是采用分析类来实现用例模型的产物。分析类包括边界类 (boundary class)、控制类 (control class)、实体类 (entity class)^[2], 分析模型的设计取决于这三种分析类的设计。分析模型是 MVC 模式的一种经典应用^[3]。MVC 模式应用程序由模型层、视图层、控制层, 分析模型中的实体类属于模型层, 边界类属于视图层, 控制类属于控制层。

由于分析模型在 RUP 中是一个可选和复杂的模型, 目前还没有一种从用例模型直接转化为分析模型简单有效方法, 所有好多软件设计师在建模时往往容

易忽略它, 只是根据自己以往的经验来进行系统的设计, 项目设计完成才发现自己的设计结果根本就不满足用户的需求, 开发项目过程中无法证明自己的设计是否符合需求。分析模型比设计模型的抽象层次要高, 它与具体的实现无关, 有助于让人们更容易理解系统, 验证用户需求很方便。另外分析模型能将模型与视图分离, 提高了系统的灵活性和复用性。而且分析模型建立完成以后, 就得到一个非常接近于设计类的模型, 可以真正进入系统设计阶段了。

2 从用例模型到分析模型

在语义上, 分析模型不代表系统的最终实现, 它是软件系统的一种高层次抽象。任何软件系统无论多复杂, 无论在什么行业, 其本质无非是人, 事, 物^[2]。人做事, 做事产生物。人驱动系统, 事体现过程, 物记录结果。所以建模的本质就是弄明白系统有什么人,

① 基金项目: 长沙环境保护职业技术学院教学科研基金(201103)

收稿时间: 2011-05-16; 收到修改稿时间: 2011-09-21

什么人做什么事,什么事产生什么物。其中,人对应用例模型的参与者,事对应用例模型的用户,物对应用例模型的分析类。

从用例模型到分析模型的步骤为:

①深入剖析用例图和用例描述,识别分析类。

② 细化用例描述中的交互活动,结合上下文分析,添加实体类的属性。

③ 在已识别出的分析类基础上,确定分析类的职责。

2.1 识别分析类

在统一软件开发过程中分析类包括边界类、实体类和控制类三种。边界类用于建立系统与其参与者之间的交互。边界类主要用来陈述和收集系统的边界需求。控制类代表协调、排序、事务处理以及对其他对象的控制,经常用于封装与某个具体用例有关的操作。实体类代表需要持续存储的信息。实体类通常用业务域中的术语命名^[4]。

对于边界类,可以从用例描述中找出与边界相关的词语,然后进行整理,包括删除重复词语。对于实体类,选择用例图中的用例名称里的名词,然后进行整理,避免出现一些大的名词和专业名词(如系统、子系统、网站、Oracle、sql、windows等)。对于控制类,有两种方法,第一种选择用例图中的用例名称里的动词,然后再进行整理,包括删除重复词语。另一种方法就是为每一个带有增删改查功能的实体类建立一个控制类,有多少这样的实体类就有多少个控制类。

2.2 添加实体类的属性

用例模型就是从用户的角度出发,考虑计算机能实现哪些具体目标,从而帮助开发人员获取更多的需求。从需求角度来说,一个用例就是用户由于某种外部事件而与计算机之间进行的一次交互^[5]。而用例描述就是要详细描述这次交互活动的过程,在交互的过程中,会产生一些结果,此结果就是分析模型中的实体类,而它的属性可以从上下文分析中找出来。

首先,从用例图里找到包含有该实体类名字的用例,然后从该用例的用例描述里找到与实体类相关的句子,结合上下文进行分析,找出属于该实体类的属性。作者在实践中总结了这种方法:如果实体类后面有“包括”之类的关键字或者有该实体类为定语的关键字,那后面的内容就是实体类的属性。如果没有,

则该实体类无属性。

2.3 确定分析类的职责或方法

确定分析类的职责常用的方法就是顺序图,该图能描述执行特定系统功能时,对象间的协作关系和消息传递顺序。根据顺序图,设计者能方便地定义相关类的动态方法。顺序图中的实线箭头相当于程序代码中一次方法调用,与之对应的虚线箭头是该方法的返回值,实线箭头所指的类则是该方法的所属类^[6]。实际上,除了上述方法以后,还有一种更简单的方法。因为用例顺序图描述了系统与环境进行交互的过程,此过程中,环境通过调用系统提供的方法与系统进行交互。所以每个用例可以被定义成一个用例控制类,与该用例相关的操作成为类的方法^[7]。

除了上述两种方法以后,还可以参考下述方法:一般来说,每个实体类至少包含一个方法,那就是它的构造函数。对于控制类来说,其方法来自两类,第一类是也是它的构造函数方法,第二类是先找到包含有该控制类名字中实体类名词的用例,然后,从该用例名称找出属于该控制类的方法。从笔者多年的实践经验来看,用例名称的动词词语就是控制类的一个方法。添加边界类方法也比较简单,其方法也来自两类,第一类是界面(窗口或页面)的初始化方法,第二类是控件引发的事件方法。

3 实例

我们以一个博客系统作为例子来说明,其管理员的用例图如下图 1:

为上述每个用例都进行基本流程描述。例如发表文章的描述是:游客在阅读全文页面中,输入评论内容(不超过 250 字),然后点击提交按钮,系统则自动保存到评论列表中,包括评论内容,评论 id 号,评论所对应的文章 id 号,评论时间,然后在阅读全文页面中的全文下面更新刚才发表的评论。(由于字数限制,其它用例的用例描述省略)

(1) 获取边界类。在用例描述中找出与窗口或页面相关的词语并整理得到:首页、阅读全文页面、文章列表页面、留言页面、搜索页面、发表文章页面、管理文章页面、文章编辑页面、文章删除页面、分类管理页面、执行 SQL 语句页面、管理员回复留言页面。

(2) 获取实体类。在用例名称中找出相关名词并整

理得到：文章、评论、留言、链接、登录、执行 SQL、文章类型。

(3) 获取控制类。为上面每一个带有增删改查功能的实体类建立一个控制类得到：文章控制类、评论控制类、留言控制类、登录控制类、执行 SQL 控制类、文章类型控制类。

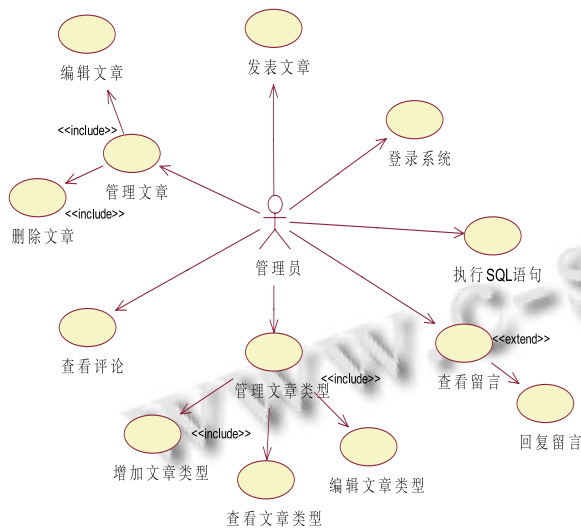


图 1 博客系统的管理员用例图

接下来，从用例描述中添加实体类的属性。包括实体类文章的用例有点击全文链接查看文章、按类别查看文章、发表文章、编辑文章、删除文章、搜索文章。而点击全文链接查看文章的基本流程是：游客进入首页，显示所有文章的标题、文章的发表时间、文章所属分类、文章的一部份内容、文章的评论数目。然后游客点击包含文章 ID 的链接进入阅读全文页面。由于字数限制，其它用例的用例描述省略。通过查看用例描述后，发现文章实体类的属性共有 7 个，分别是标题、发表时间、所属分类、内容、评论数目、文章 ID、类别 ID，通过建模工具 Rose 2003，画出其类图如下图 2，再根据上面 2.3 节所述，文章实体类（Article 类）的操作方法就是其构造函数 Article()，其类图如下图 3。

而对于边界类 article 类来说，首先直接在其类图里添加窗口或页面初始化方法(新建一个页面或窗口时，开发工具一般自动创建这个方法，不需要开发人员手动创建，如 vs2008 窗口初始化方法是类名_Load(),页面初始化方法是 Page_Load()。然后，在用例的文字描述中找出相应的控件（比如文字描述中有点击某个按钮，则按钮

就是一个控件），记录其事件方法。我们发现提交事件（ btnCommit_ServerClick ）和重置事件（ btnReset_ServerClick ）。最后，画出其类图，如图 5。

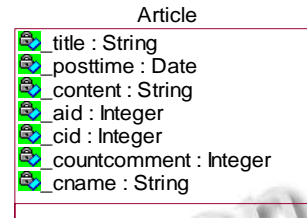


图 2 Article 类（文章实体类）的属性类图

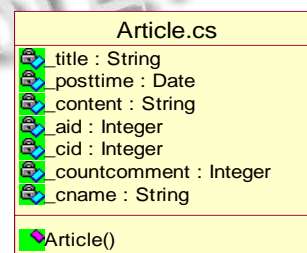


图 3 Article 类（文章实体类）的方法类图



图 4 ArticleOperate.cs 类（文章控制类）的方法类图

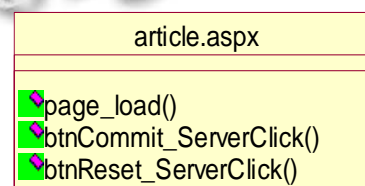


图 5 Article.aspx 类（阅读全文页面边界类）的方法类图

3 总结

在统一软件开发过程 RUP 中，从用例模型过渡到分析模型是其中的关键环节，也是让很多程序员感到很困难的一件事，作者经过多年的编程实践，总结了这种方法，它不但简单，而且很实用，能显著提高程序员的软件开发效率。

(下转第 182 页)

KVM 线程号为 3906, 异步 IO 事件处理线程号为 4212, 如图 6 所示。此时图 4 中线程号为 3910 的 aio 线程消失, 动态产生 aio 线程 4212。

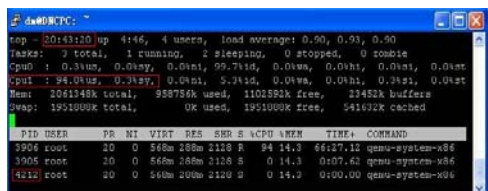


图 6 t2 时刻 KVM 虚拟机进程运行状态

根据 Linux 内核的 PID 命名规则可知, PID 逐渐递增, aio 线程的产生是一个动态过程, 根据 Guest OS 的需要而产生。从而进一步证实, 如果用采用方案一, 则难以实现 aio 线程的绑定。

经过长时间的实验观察, 虽然运行 Guest OS 时 CPU 核 1 的利用率达到 94.7% 左右, KVM 虚拟机始终保持在指定的核 1 运行, 即使核 0 十分空闲, 仍不会发生迁移。可以通过 taskset 命令查看这三个进程的 Affinity 值, 得到进一步的验证, 如图 7 所示。图中 3905, 3906, 4212 进程的 Affinity 值为 2 (即 0x00000010), 表示相应进程或线程运行在核 1 上。

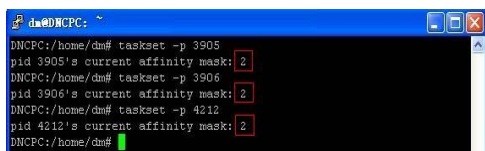


图 7 用 taskset 命令查看虚拟机进程的 Affinity 值

(上接第 193 页)

参考文献

- 1 克鲁森. Rational 统一过程实践者指南. 徐正生译. 北京: 中国电力出版社, 2004. 260.
- 2 谭云杰. Thinking in UML. 北京: 中国水利水电出版社, 2009. 15.
- 3 陈丽娟. 一种用例驱动的应用系统分析与设计方法. 计算机应用与软件, 2002, 19(4), 42-44.
- 4 任兴来. 基于用例的用户界面原型设计研究. 青岛: 青岛大

5 结论

本文通过对 KVM 虚拟机的绑定, 实现了 KVM 虚拟机处理器核之间的隔离, 使 KVM 虚拟机在指定的物理 CPU 核上运行, 不发生核间的迁移, 从而达到物理 CPU 资源的指定使用。当运行多个 Guest OS 时, 只需在运行时指定 CPU 参数, 即可让各个 Guest OS 始终运行在指定的核上, 进而能够充分有效地利用计算机硬件资源^[8]。

参考文献

- 1 The Intel®64 and IA-32 Architectures Software Developer's Manual. 2009: 10-110.
- 2 KVM 官方网址 http://www.linux-kvm.org/page/Main_Page.
- 3 QEMU 官方网址. http://wiki.qemu.org/Main_Page.
- 4 Smith JE, Nair R. Virtual machines. Publishing House of Electronics Industry, 2005: 3-10.
- 5 Qumranet Inc. KVM: Kernel-based Virtualization Driver. 2006: 1-5. http://www.linuxinsight.com/files/kvm_whitepaper.pdf.
- 6 李胜召, 郝沁汾, 肖利民. KVM 虚拟机分析. 计算机工程与科学, 2008, 30(A1): 129-130.
- 7 英特尔开源软件技术中心. 复旦大学并行处理研究所. 系统虚拟化. 北京: 清华大学出版社, 2009. 104-128.
- 8 Eddolls T. VM performance management McGraw-Hill Book Co., 1989: 5-10.

学, 2009.

- 5 孔军, 等. 基于 UML 的系统需求分析. 计算机工程与应用, 2003, 15: 217-218.
- 6 方红萍, 陈和平. 信息系统 UML 建模研究. 计算机工程与设计, 2006, 27(19): 3614.
- 7 陈鑫, 李宣东. 基于设计演算的形式化用例分析建模框架. 软件学报, 2008, 19(10): 2548-2549.