一种用例驱动的需求关注点分离的方法

韩晓英,虞慧群
(华东理工大学计算机科学与工程系,上海 200237)

摘 要: 用例驱动技术从行为者的角度建立用例,旨在减少软件开发的复杂度.然而,由于 OO 技术的不足,不同用例之间存在关注点的横切与缠结,从而使得软件开发更加复杂.本文提出了一种用例驱动的方法——UCD/ Theme 方法.该方法利用面向方面技术实现关注点在需求阶段的分离.ATM 案例分析展示了该方法的可行性.

关键词: 面向方面; Theme; 用例驱动; 关注点分离
中图分类号: TP311 **文献标识码**: A **文章编号**: 0372-2112 (2006) 12A-2498-04

A Use-Case Driven Approach to Supporting Separation of Concerns on Requirements

HAN Xiao-ying, YU Hui-qun
(Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, China)

Abstract: Use-case driven development aims at reducing software complexity through use case that starts with the actor. However, some tangling and scattering concerns in different use cases look tricky because of the deficiency of object-oriented technique. It makes software more complex. This paper presents a use-case driven extension approach-UCD/ Theme approach, which uses aspect-oriented technique to facilitate separation of concerns on requirement analysis phase. Finally an ATM case study is given to illustrate the feasibility of the method.

Key words: aspect-orientation; theme; use-case driven; separation of concerns

1 引言

现在,大部分软件开发商利用面向对象方法[1]解决软件开发过程中关注点的分离问题,但是在处理复杂系统时存在一定的不足.用例驱动分析(Use-Case Driven Analysis,简称 UCDA)通过 actor-use case 视图,从行为者的角度出发建立用例,在复杂的需求分析中起到了巨大的作用[2].从而使得软件开发从以代码为中心转为以用例为中心.

用例和用例驱动开发的概念一经 Jacobson 首次提出,就被成功应用到许多软件项目中,已成为捕获关注点的一个标准方法[3].但是,定义在对象基础之上的 UCDA 存在一些问题有待解决:

- 大部分用例不是独立的,它们可能互相重叠,同时发生,互相影响.
- 即使两个或多个用例彼此独立,但其中有些方法或操作执行相同动作或对系统有相似影响.
- 一些非功能需求,如日志、实时、安全等横切用例和操作.
- 一些用例拥有一个以上的功能,而这些功能是分散在不同的用例中.

更多关于 UCDA 的信息,请参考文献[2].

面向方面的软件开发方法(AOSD)[4,5]的基本思想是对软件的非功能模块和功能模块分别独立设计,然后系统地将他们编织成完整的软件模块,从而克服了传统面向对象方法的不足.本文将面向方面方法引入到 UCDA 中,提出了一种用例驱动的面向方面软件开发方法 UCD/ Theme (Use-case Driven/ Theme).通过应用面向方面的技术在需求分析阶段保持关注点的完全分离[6],改进 UCDA 技术,并且一直延续到设计、编码阶段,从而使软件开发具有更高的可追踪可重用性和有效性.

另一个相关的工作是 Clarke 和 Baniassad 的 Theme 方法[7].该方法在 UML 中引入了 theme 的概念,在一定程度上体现了面向方面的思想.与 UCD/ Theme 不同之处在于,Theme 方法中的需求是作为一系列的纯操作进行分析的,在组合与验证时没有一个清晰明确的标准.若需求有所变化,则难于利用这种方法进行软件演化,大多要影响到其它的需求,而这些需求需要逐个查找.本文提出的 UCD/ Theme 方法具有更高的灵活性.

收稿日期:2006-05-20;修回日期:2006-12-15
基金项目:国家自然科学基金(No. 60473055);上海市浦江人才计划(No. 05PJ14030)

2 UCD/ Theme 方法

2.1 Theme 的语义

Theme 方法中的概念 theme 被定义为"关注点的封装",它不等同于 AO 中的 aspect,而比 aspect 更通用,描述一系列关注点的集合. UCD/ Theme 在此基础上引入了用例驱动技术,从而扩展了 theme 方法,使其更易于与现有技术集成.

UCD/ Theme 方法中的 theme 与 Theme 方法中的 theme 具有不同的含义. UCD/ Theme 中的 theme 是用例中一个步骤的概括,可能是一个操作或一个关注点,这里我们可以用"pure use case"来描述,是对 use case 的更细粒度的描述.

2.2 分析视图

在 UCD/ Theme 方法中,有四种视图用来进行需求分析: actor-use case 视图,theme-use case 关系视图,crosscutting 视图以及 pure use case 视图:

- Actor-use case 视图:UML 中 use case 之间有三种关系:扩展(extend)、包含(include)以及继承(generalization). 在 actor-use case 视图中这三种关系都要表示出来,从 actor 的角度出发捕获初始需求.
- Theme-use case 关系视图:如果在一个 use case 描述中提到了某个 theme,则这个 theme 和相关的 use case 之间就有一个连接. 给定一组 use case 用例描述和一组 theme,即可生成.
- Crosscutting 视图:从 actor-use case 视图中可以看到,有些 use case 是与多个 use case 相关联的. 同样,在 theme-use case 关系视图中,一些 theme 与多个 use case 相关联. 我们将这种分散的关注点(分散的 use case extension 和分散的 theme)称为横切关注点. 这里,为了与 UML 标准中 use case 扩展表示相统一,我们用虚线箭头表示从 aspect theme 到 use case 的横切关系.
- Pure use-case 视图:pure use-case 有两种:base use-case 和 aspect use-case. 无论是哪种 pure use case,我们用一个 pure use-case 视图来表示仅与单个 pure use-case 相关的一些需求信息.

2.3 分析方法

UCD/ Theme 方法中的需求分析阶段,即寻找出系统相关的 pure use-case,概括起来,有五个主要的基本活动,每个活动结束会产生相应的视图,如图 1 所示.

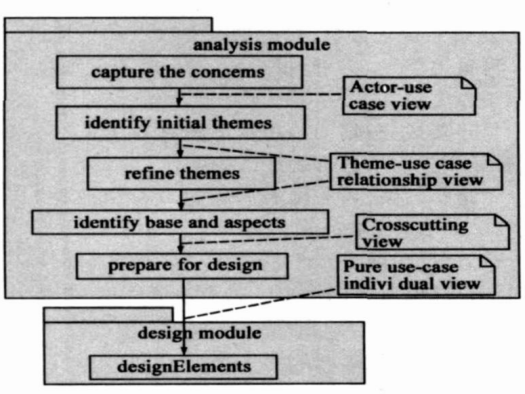

图 1　UCD/Theme 中的活动与视图关系

3 案例分析

这里我们用 ATM 自动提款机这个典型案例[8],说明如何应用 UCD/ Theme 方法的五个步骤进行需求分析.

3.1 捕获关注点

第一步是利用 OO 的 use case 捕获系统关注点. 从客户的角度看,这里有三个 base use-case,一个 extended use-case. 我们用 actor-use case 视图描述这种关系,如图 2 所示.

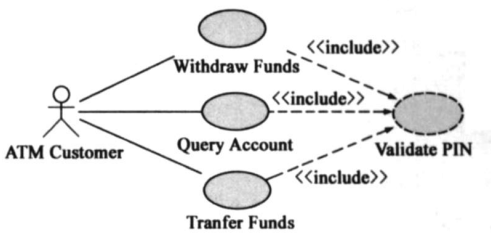

图 2　Actor-use case 视图

3.2 定义初始 theme 集

从第一步可以看到,有三个 base use-case,这里我们用提款(withdraw funds)这个用例为代表说明如何应用 UCD/ Theme 方法进行分析,其余的用例可分析类似. 提款用例的描述如图 3.

在图 3 中,我们用一个黑体词表示 Withdraw funds use case 中的每一步. 然后可以列出一些初始 theme: elect, enter, check, confirm, pay, print, return, play. 同样地,我们可以从查询用例(Query Account)和转帐用例(Transfer Funds)中得到其它的一些 theme. 这样初始 theme 集为:{select, enter, check, confirm, pay, print, return, display, read}.

List1: Withdraw Funds specification
Basic flow: The cardholder **selects** withdraw funds option; The cardholder **enters** the desired withdrawal amount; The ATM **checks** the whether there are enough amount in the account and desired amount against the daily withdrawal limit; If check passed, the ATM **confirms** to pay amount; The ATM **pays** the amount to the cardholder; The AFTM **prints** a receipt; The ATM **displays** "withdraw OK" message; The ATM **returns** the card to the cardholder; The ATM **displays** "Welcome" message.
Alternative flow: 1. If the accountNo inputted is illegal, the ATM **displays** "error" message; The ATM **returns** the card to the cardholder; 2. If the amount requested is greater than the daily withdrawal limit, the ATM **displays** "sorry" message; The ATM **returns** the card to the cardholder; 3. If there are not enough money in the system, the ATM **displays** "sorry" message; The ATM **returns** the card to the cardholder.

图 3　提款用例描述

3.3 精化 theme 集

初始 theme 集合中有十个 theme,但有些 theme 表意是不清楚的. 我们用 Theme 方法中的四个操作来进行精化: add,

split up ,group ,以及 delete. 比如 ,“enter ”这个 theme 根据不同的选择要输入不同的信息 ,所以 ,我们用 enter-amount 和 enter-account 来代替 enter. 我们还可以看到 ,当 display 发生时 ,return 也相应地发生 ,也就是说 ,这两个 theme 是不可分割的 ,所以 ,我们可以将它们合并为一个新的 theme : end. 另外 ,我们的 UCD/ Theme 方法中还增加了另一种 rename 操作以利于理解. 在执行完这五个操作后 ,最终的 theme 集为 : { select , enter-account , enter-amount , checkWithdraw , checkTransfer , confirm , pay , print ,readBalance ,end} .

这样我们就可以画出 theme-use case 的关系视图 ,如图 4 所示.

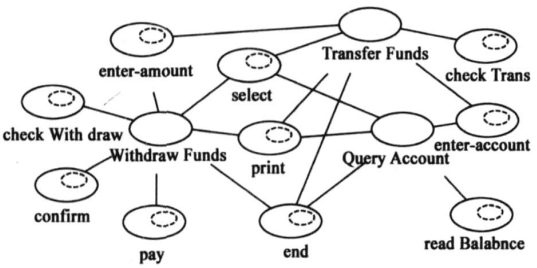

图 4 Theme-use case 关系视图

3.4 定义 base 和 aspect

我们可以从两个视图中定义 aspect : actor-use case 视图和 theme-use case 关系视图.

在 actor-use case 视图中有三种用例关系 ,被扩展出的用例 ,被包含的用例 ,以及被继承的用例都可以视为 aspect use case. 例如 , Validate PIN use case 被三种用例包含 : With draw Funds、Query Account 以及 Transfer Funds ,所以将 Validate PIN 用例作为一个 aspect use case.

Theme 和用例之间的关系是多对多的关系. 有些 theme 与多个用例相关 ,我们将之视为一个 pure use case ,这些 theme (pure use case) 就是用例之间的一个横切关注点 ,所以我们称其为 aspect use case. 图 5 表示了本例中的 aspect use case.

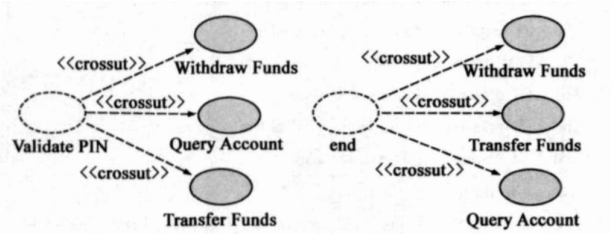

图 5 Crosscutting 视图

3.5 准备设计

为了利于从需求分析到设计的过渡 ,我们将每个 use case 单独表示出来 ,pure use case individual 视图为我们解决了这个问题. 在这个视图中 ,所有与这个 pure use case 相关的元素都表示在一个视图中 ,从而有利于进行后阶段的设计. 所以 ,我们需要定义 pure use case 中的 objects.

我们在识别出的 entity 和相关用例之间用一个关联关系来表示 ,其中灰色的 entity 表示“共享 entity ”,即在多个 pure use case 中都会出现. 如图 6 所示. 同时我们将横切关系也表示出来.

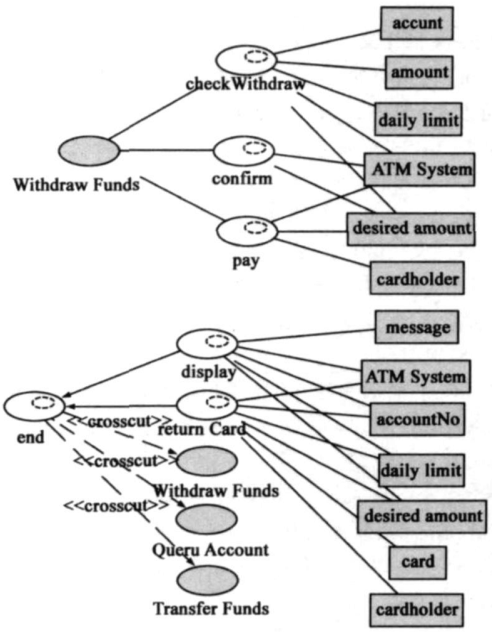

图 6 Pure use case 视图

4 ATM 实时要求

在前面的设计中 ,我们没有考虑系统的实时性. 这里我们在需求中加入时间特性 Time- Limit 来限制系统的响应时间. 在分析的各阶段的结果视图中所作的改动如图 7 所示.

从图中可以看出 ,在加入时间特性后 ,我们仅需要在现有的分析模型上作少量的改动 : 在 actor-use case 视图中增加 time aspect ,在 Theme-use case 关系视图中增加 time-aspect ,在 cross-cutting 视图和 pure use case individual 视图中分别增加相应的 time-aspect 部分. 大大减少了对软件的修改.

5 结论

大规模的软件开发包括一些参与者 ,每个参与者可能有许多不同的角色、任务以及关注点 ,这些随时都可能会发生变化 ,这就可能在后续阶段产生一些冲突. 所以 ,在早期阶段进行关注点的分离是非常重要的[9]. 本文介绍了一种 UCD/ Theme 方法进行需求分析 ,总的来说有以下几个优点 :

· 可重用性高. 使用 pure use case 来表示细粒度的关注点 ,可用于多个模块.

· 易于集成. 根据初始用例来集成 pure use case ,并且易于根据初始用例需求进行验证.

· 可追踪性强. 当需求有所差错或有所改动时 ,可快速追踪到相关的 pure use case.

· 有效分离. 通过五个基本活动分析出 pure use case ,可单独进行设计 ,最后再根据组合关系集成起来. 并行开发提高了开发效率.

UCD/ Theme 方法应该是一个面向方面的软件开发方法 ,而不仅仅是一个分析方法. 我们将继续研究 pure use case 的设计、实现和验证工作.

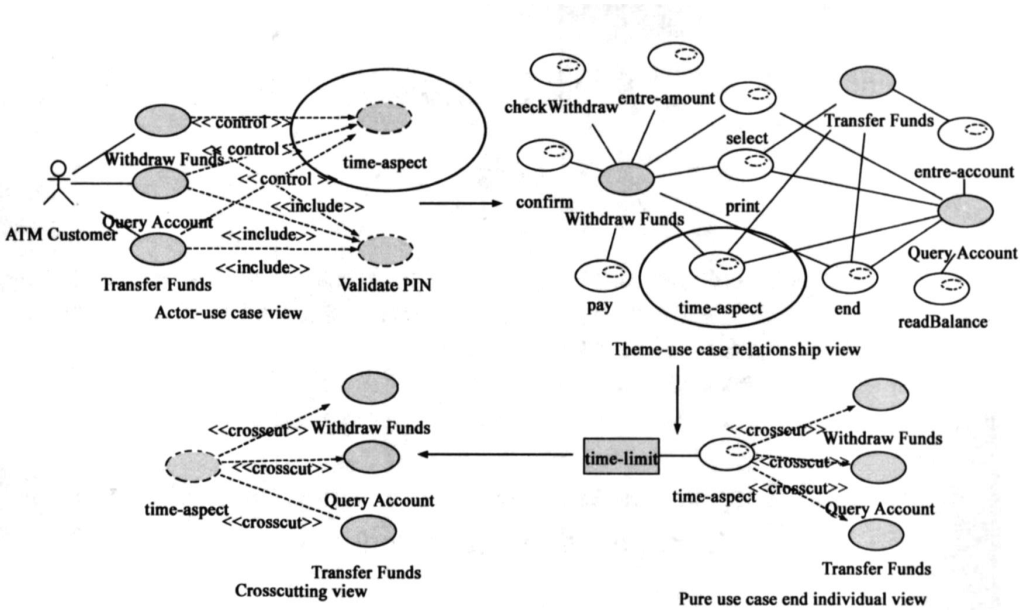

图 7　需求中增加 time-aspect方面

参考文献:

[1] J Rumbaugh, I Jacobson, G Boosh. The Unified Modeling Language Reference Manual[M]. Boston, Massachusetts: Addison-Wesley, 1999.

[2] B Regnell, K Kimbler, A Wesslén. Improving the use case driven approach to requirements engineering[A]. In Proceeding of second IEEE International Symposium on RE'95[C]. York: IEEE, 1995. 40 - 47.

[3] I Jacobson, P W Ng. Aspect-Oriented Software Development with Use Case[M]. Boston, Massachusetts: Addison-Wesley, 2004.

[4] G Kiczales, J Lamping, et al. Aspect-oriented programming [A]. In Proceedings of the European Conference on Object-oriented Programming[C]. LNCS 1241, Berlin: Springer-Verlag, 1997. 220 - 242.

[5] R France, I Ray, G Georg, S Ghosh. Aspect-oriented approach to early design modeling[J]. IEE Proc. Software, 2004, 151 (4): 173 - 186.

[6] A Solberg, D Simmonds, R Reddy, S Ghosh, R France. Using aspect oriented techniques to support separation of concerns in model driven development[A]. In Proceedings of the 29th Annual International Computer Software and Applications Conference[C]. Edinburgh, Scotland, 2005. 25 - 28.

[7] E Baniassad, S Clarke. Theme: An approach for aspect-oriented analysis and design[A]. In Proceedings of the 26th ICSE[C]. 2004, 158 - 167.

[8] H Gomaa. Design Concurrent, Distributed, and Real-Time Application with UML[M]. Boston, Massachusetts: Addison-Wesley, 2000.

[9] J Araújo, P Coutinho. Identifying aspectual use cases using a viewpoint-oriented requirements method[A]. In Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design, Workshop of the 2nd International Conference on Aspect-Oriented Software Development[C]. Boston, USA 2003.

作者简介:

韩晓英　女,1982 年 11 月出生于山西忻州,2004 年 9 月开始在华东理工大学计算机科学与工程系攻读硕士学位. 主要研究方向:软件工程、面向方面.

E-mail: hanxiaoying82@yahoo.com.cn

虞慧群　男,1967 年出生于江苏溧阳,教授、博士生导师、IEEE 计算机学会会员、ACM 会员、中国计算机学会高级会员,1995 年获上海交通大学工学博士学位. 现为华东理工大学计算机科学与工程系主任,主要从事高可信计算、软件工程、形式化方法及应用等方面的研究. E-mail: yhq@ecust.edu.cn