基于本质用例的软件需求分析和获取*)

吴 斌 汪 琦 顾 庆 陈道蓄
(南京大学软件新技术国家重点实验室 南京210093)

摘 要 本质用例描述了用户与系统间抽象、轻量级和技术无关的对话过程。与传统的用例相比,本质用例更为简洁,避免了在开发过程早期就进行设计决策。本文探讨了在软件系统开发过程中,使用本质用例进行需求分析与获取的过程,并将其与传统的用例技术进行了比较。

关键词 软件需求,用例,本质用例,用例图,用例对话,角色扮演

Essential Use Case Based Software Requirements Analysis and Elicitation

WU Bin WANG Qi GU Qing CHEN Dao-Xu
(State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093)

Abstract Essential use cases are abstract, lightweight, technology-free dialogues between user and system. Compared with conventional use cases, the steps of essential use cases are more concise. The benefit of this abstraction is that no particular design decision needs to be made at the early stage of system development. This paper discusses an approach of software requirements analysis and elicitation based on essential use case, and compares it with the conventional use case technology.

Keywords Software requirement, Use case, Essential use case, Use case diagram, Use case dialogue, Role-play

1 引言

软件需求分析和获取是软件系统开发过程中最为困难也是最为关键的部分,只有真正满足用户需求的软件产品才能为用户所接受。根据 Leffingwell 在1997年所做的研究,软件项目中40%～60%的问题都是在需求的分析和获取阶段埋下的祸根。1992年 Jacobson 在其著作《Object-Oriented Software Engineering: A Use Case Driven Approach》中首次提出用例(Use Case)的概念[1],用于软件需求的分析与获取。用例以其独有的优越性,获得越来越多的软件系统开发者的青睐,并受到业界标准建模语言 UML 和建模过程 RUP 的广泛支持。

随着使用的深入,用例的局限性也逐渐体现出来。正如 Larry Constantine 和 Lucy Lockwood 所提出的那样:"传统的用例包含了太多有关用户界面设计的内建假设,而这些假设往往是隐含和难于理解的[2]。"这样便导致在系统设计早期就必须进行某些设计决策,并将其纳入需求规格说明,从而难于变更以适应随后的变化。

人们开始认识到在用例建模中需要"以目的为中心(Purpose-Centered)"的方法,以及在用例构造过程中将重点转移到抽象[3]。Kaindl [1995]和 Cockburn [1997; 2000] 建议将目标合并到用例中,Graham [1996]讨论了抽象用例的价值。至少在概念上,统一的观点正在形成。在这种环境下,本质用例(Essential Use Case)的概念应运而生。

本质用例最初由 Larry Constantine 和 Lucy Lockwood 在文[2]中提出,其主要目的是为用户界面设计获取需求。Robert Biddle 等在此基础上提出将本质用例应用到整个面向对象的软件开发过程中[7]。本文着重讨论了将本质用例应用于更为普遍的软件需求分析和获取过程,并结合实例讨论了该方法相对于传统用例技术所独有的优越性。

2 本质用例

2.1 用例

Jacobson 在他1992年的著作中将用例定义为"与系统进行对话时行为相关的事务系列"[1]。在最近的 RUP 定义中,对用例的定义没有实质性的改变,认为用例是"一系列包含变量的动作描述,系统由此对特定用户产生有价值的可见结果"[4]。

关于用例的一种更为普遍的观点是:用例展示了系统与外界的一系列有意义的交互过程。

在软件开发的早期,用例着眼于交互以提取系统行为,从而帮助开发者获取系统需求、确定系统规范。用例技术使用格式化的语言和图形对系统的交互过程进行描述,易于理解,尤其适用于在大范围内(包括终端用户,项目风险承担者,以及其他没有直接系统开发经验的用户)进行需求采集和分析工作。

在后续的开发阶段中,用例可以帮助开发者对需求进行有效的划分,并通过组合、过滤、设定优先级等方式组织需求,协助管理整个开发过程。

2.2 本质用例

本质用例是"以使用为中心设计"的一部分,由 Larry Constantine 和 Lucy Lockwood 提出[2]。Constantine 和 Lockwood支持用例,并认同用例的很多显著优点。但他们同时

*)基金项目:国家863高技术项目"基于 CMM 的质量保证平台及应用"(编号:2001AA113090)。吴 斌 硕士研究生,研究方向:软件需求工程,工作流系统。汪 琦 硕士研究生,研究方向:软件需求工程,工作流系统。顾 庆 博士,副教授,研究方向:分布式计算,工作流系统。陈道蓄 博士生导师,研究方向:分布式计算与并行处理。

注意到传统用例所具有的局限性:包含过多的有关用户界面设计的内建假设,导致在系统设计早期就必须进行某些设计决策,给后续的开发工作带来不便。

本质用例正是为了克服上述问题所提出的。术语"本质"来源于:本质模型是"通过与技术无关的、理想化的、抽象的描述来捕捉问题的实质"。Constantine 和 Lockwood 对"本质用例"的定义如下[2]:

"本质用例是使用应用领域或用户语言的结构化叙述,包含对某一任务或交互的简明扼要、技术无关、独立与实现的描述。该任务或交互从用户的视角来看是完整的,有意义的和良好定义的,并能体现系统的目的或意图。"

2.3 本质用例的表示

本质用例用一种格式化的方法来记录用户和系统的对话过程。这种格式与 Wirfs-Brock 用于表示用例的两列表类似[5]。在 Wirfs-Brock 的格式中,用"用户动作"和"系统响应"作为两列的标识。尽管 Wirfs-Brock 讨论了用例的不同抽象层次,但它所表示的两列用例包含的是用户与系统交互的具体步骤。

相应地,在本质用例中采用"用户意图"和"系统职责"作为两列的标识。这两个新的标识表明:本质用例通过在记录交互时隐藏具体的实现细节来支持抽象。需要注意的是,这个抽象过程并非与整个用例相关,而是与用例的各个步骤相关。通过这种方法,本质用例指定了一个由抽象步骤所组成的交互序列。

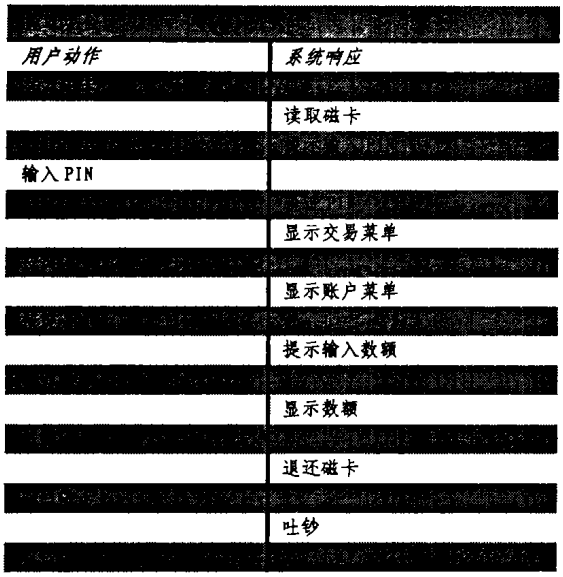

图1 从自动取款机取款的传统用例表示

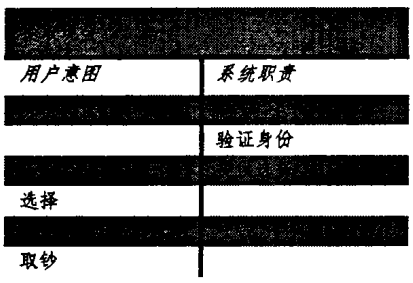

图2 从自动取款机取款的本质用例表示

图1和图2所示的例子由 Constantine 和 Lockwood 所给出[2]。图1为传统用例,用"用户动作"和"系统响应"进行描述;图2为本质用例,用"用户意图"和"系统职责"进行描述。可以看到,本质用例的步骤更为抽象,允许多种具体实现。它更简短而且易于理解。

3 关于使用本质用例的讨论

传统用例作为一种软件需求分析与获取的手段,其有效性已经得到了广泛的认可,并得到越来越多的建模语言和开发过程的支持。在用例技术日益完善的今天,我们是否有必要引入本质用例,用以取代传统用例在需求的分析与获取过程中的地位呢?本部分,我们将通过传统用例与本质用例的比较,对该问题进行较为深入的探讨。

本质用例的概念源于传统用例。可以说,它继承了传统用例在软件需求分析与获取方面的许多优势。同传统用例一样,本质用例关注用户与系统之间的交互过程。Wirfs-Brock 指出,用例可以看作用户和系统的"会话(Conversations)",这种形式有助于过程建模。同时,对交互的强调还可以帮助开发者确定系统边界,划清系统内部和外部的界限。

与传统用例不同的是,本质用例更加强调"以目标为中心"。相对于传统用例,本质用例的步骤更为抽象,而这种抽象性所带来的好处是显而易见的:简短的卡片式描述可以加速分析过程;抽象使我们无需在系统开发的早期就对交互的细节问题纠缠不清;我们可以集中精力确定系统的本质用例,从而避免被系统的具体实现所困扰而耗费大量时间。

在描述用户与系统的交互过程时,本质用例采用术语"用户意图"和"系统职责",而非传统用例的"用户动作"和"系统响应"。

3.1 用户意图

为了获取需求,开发者必须努力把自己融入用户的角色,并以用户的视角来思考问题。本质用例中对用户意图的强调无疑强化了这一方面。为了理解用户意图,开发者必须了解用户的特点以及他们在系统中所处的地位,并对用户的动机做深入的研究。这使得所获取的用例更具有说服力,同时也有利于对用例的完备性和一致性做出评价。

传统用例关注系统与外界的交互,尤其是系统的使用方式。然而在大多数的系统设计中,使用方式是不能预先确定的。这就给用例的确定带来了困难。本质用例同样是以使用为中心,但可以在理解用户意图的基础上确定系统的使用方式,从而进一步确定用例本身。

在诸如 RUP 等软件开发过程中,为了理解用户及其需求,在早期就对用户界面原型进行开发。在基于本质用例的开发过程中,通过对用户意图的理解,可以为用户建立一个清晰的流程思想。这是一种轻量级的方法,不需生成具体的用户界面原型,因而支持更快速的开发过程。

3.2 系统职责

引入用户意图是为了描述用户需要实现的目标,而引入系统职责是为了描述系统必须履行的责任。

需求获取的一个重要方面是确定系统边界,即搞清楚系统到底要做什么。本质用例中对系统职责的强调有助于系统边界的确定。类似于在面向对象的设计中所采用的方法,开发者可以将整个系统看作一个"黑盒",而用系统职责来描述系统的行为。

系统职责描述系统需要做什么,而不是系统具体如何实现。在基于本质用例的开发过程中,通过抽象,使我们无需在早期就对用户界面进行设计,并且整个开发过程与技术无关。

从用户的角度，抽象使我们不必考虑意图的表达方式；从系统的角度，抽象使我们不必考虑职责的实现细节。

传统用例强调人机交互中系统对用户的响应。然而在通常的系统开发中，系统的功能并不只限于此。如果仅仅关注系统对用户的响应，则很有可能遗漏重要的系统功能。在本质用例中，通过对系统职责而不是系统响应的关注，可以较好地解决这个问题。

通过以上的分析，我们不难看出：在软件需求的分析与获取过程中，本质用例具有很多独到的优势，是传统用例所无法比拟的。因此，我们对基于本质用例的需求分析与获取的研究，有着很高的实用价值。

4 基于本质用例的软件需求分析与获取

本文旨在使用本质用例进行软件需求的分析与获取。我们希望该方法对那些没有任何技术背景的用户来说更加容易理解，并使得他们更为有效地参与到软件需求开发的活动中。本质用例的许多特性刚好满足我们的希望。

本部分我们将使用一个小型的艺术中心订票系统（Arts Center Booking System，以下简称为 ACBS，该系统在文[6]中有简单描述），以此为实例说明基于本质用例的需求分析与获取过程中的主要技术细节问题。

4.1 本质用例的确定

首先介绍如何从系统中确定本质用例。该过程可以分为以下4个步骤：①确定执行者（Actor）；②列出候选用例（Candidate Use Case）；③选择焦点用例（Focal Use Case）；④绘制用例图（Use Case Diagram）。

4.1.1 确定执行者 定义系统最重要的任务之一是搞清楚系统到底是什么。我们通过确定系统的执行者——直接与系统交互的外部实体，包括用户及其他外部系统——来达到这个目的。

首先，对客户及其行为进行调研分析，确定真正使用系统的用户列。然后，针对列中的每类用户，理解其使用系统的目的或意图，并以此对执行者进行分类。最后，根据执行者相对于系统的重要性对其大致划分优先级。

通过对 ACBS 系统的分析，我们可以确定该系统的执行者主要有以下几类：

售票员：主要负责售票，座位预订，接受顾客咨询等事宜；

业务经理：主要负责场馆预订，表演安排，座位管理等事宜；

商务经理：主要负责上座率统计，营业额汇报等事宜；

会计系统：作为与 ACBS 系统相接的外部系统，主要负责营业额的统计。

4.1.2 列出候选用例 接下来要做的是确定系统到底要做什么。这可以通过列出每类执行者所对应的候选用例来实现。

用例描述了执行者与系统之间的一个单一的交互序列。从执行者的观点来看，系统应该是一个不包含任何内部实现细节的“黑盒”。因此，候选用例应尽可能的简洁明了。可以运用领域知识，文本分析等手段协助候选用例的发掘。

针对 ACBS 系统的执行者，我们列出以下候选用例：

售票员：售票，退票，提供座位信息（剩余情况，具体位置，价格），预定座位，取消座位预订，获取节目单……

业务经理：添加节目，删除节目，安排表演时间，更改表演信息，制定座位计划，取消座位计划……

商务经理：上座率汇报，月营业额汇报，年度营业额汇报……

会计系统：月营业额统计，年度营业额统计……

4.1.3 选择焦点用例 候选本质用例的粒度相当小，但数量可能很大。一般来说，一个小型系统可能包含40～50个候选用例，而一个中型系统则可能达到200～300个之多。如何有效地管理这些候选用例，并为其划分优先关系？我们通过选择焦点用例来解决这个问题。

很难确定到底哪些用例是“重要”的，因而我们使用术语“焦点”：以该类用例为焦点驱动整个开发过程。我们可使用以下方法选择焦点用例：针对用例的使用频率，开发风险等方面对其评分，将各方面的得分相加，然后将用例按总分排序。取总分最高的10%的用例（最多20个）作为焦点用例。以 ACBS 系统为例，我们得到如下的焦点用例：售票员：售票，预定座位，获取节目单；业务经理：添加节目，安排表演时间；商务经理：上座率汇报，月营业额汇报；会计系统：月营业额统计。

4.1.4 绘制用例图 如何确定用例的完备性？换句话说，如何得知已经拥有足够的用例？绘制用例图是一个行之有效的方法。如果开发的系统包含超过20个用例，最好为每类执行者（或几类相关的执行者）绘制一张用例图。ACBS 系统的用例图如图3所示。

针对每张用例图，考虑以下方面：(1)是否每类用户都有相应的执行者与其对应；(2)是否每类执行者通过相关用例都能实现自己的意图或目标；(3)是否已包含所有明显用例且没有遗漏。如果对上述问题的回答都是肯定的，则可以认为用例是完备的。

4.2 本质用例的细化

通过以上的步骤，我们已经得到了一系列的候选用例。接下来所要做的就是对它们进行细化。细化的过程可分为以下两步：①构造本质用例对话过程；②用例角色扮演。

4.2.1 构造本质用例对话过程 目前所得到的只是一系列的用例名，需要了解有关本质用例的更多细节，诸如执行者需要对系统提供哪些信息？为了实现用例系统需要提供哪些功能？针对上述问题，我们为每个用例构造本质用例对话过程。

我们使用用例卡（Use Case Cards）来记录本质用例对话过程。用例卡的形式与前文所述的本质用例的表示形式类似：每张用例卡针对一个本质用例，包含用例名以及相应的对话步骤。用例卡分为左右两部分。左侧为“用户”区，关注用户的实际意图或目标；右侧为“系统”区，强调系统的职责所在；中间的分隔线可以看作是用户与系统之间的接口。

针对 ACBS 系统的本质用例“添加节目”，其用例卡如图4所示。

添加节目	
指定节目单	
指定新节目细节	
	确定新节目单

图4 本质用例“添加节目”的用例卡

4.2.2 用例角色扮演 怎样保证本质用例对话过程的正确性和一致性？我们所给出的方案是：在开发小组中进行用

（下转第147页）

不足，本文提出一个扩展 LSM 框架的方法，以增强 LSM 框架对审计机制的支持能力。论文通过对 LSM 的扩展，讨论了审计功能的实现方法，论述了如何在 LSM 框架中加入审计钩子及如何在内核函数中插入钩子函数。依照这种方法，作为安全操作系统 SECIMOS 开发工作的一个重要组成部分，我们在 Linux 内核中实现了一个审计系统，取得了很好的效果。

参 考 文 献

1 訾小超，姚立红，曾庆凯，茅兵，谢立. 操作系统安全增强技术研究进展. 高技术通信，2003

2 Morris J，Smalley S，Kroah-Hartman G. Linux Security Modules: General Security Support for the Linux Kernel. In Linux Security Modules: General Security Support for the Linux Kernel，2002

3 Edward G A. Fundamentals of computer Security Technology. Prentice-Hall International Inc. 1994

4 国家863计划信息安全技术主题课题验收文件，操作系统与数据库平台安全核心技术研究技术报告. 中国科学院软件研究所，2004. 3

5 中华人民共和国国家标准. 计算机信息系统安全保护等级划分准则 GB17859-1999. 中国，1999-9-13

6 陈慧，石文昌，梁洪亮，孙玉芳. 操作系统内核级安全审计系统的设计与实现. 计算机科学，2004，31(8)

8 McVoy L，Staelin C. lmbench: Portable tools for performance analysis. In: Proc. Winter 1996 USENIX，San Diego，CA，Jan. 1996. 279～284

(上接第143页)

例角色扮演。

用例角色扮演的过程如下：指定两个开发小组成员，分别扮演用户和系统两个角色。他们以本质用例对话过程为脚本，模拟用户与系统之间的实际交互。其他的开发小组成员则作为观众监督整个角色扮演过程。

在监督过程中需要注意以下问题。首先是连续性：确保用户和系统都能正确理解对方意图并作出适当的响应，不会因为误解而造成交互过程的中断；其次是避免假定信息：有时用户或系统会假定自己已获得某些信息，而实际上这些信息并未被提供。这两种情况一定要注意避免。

可以让项目风险承担者和领域专家共同参与到监督过程中来。整个角色扮演过程重复执行，直至整个开发小组对用例对话过程的理解都达成一致。

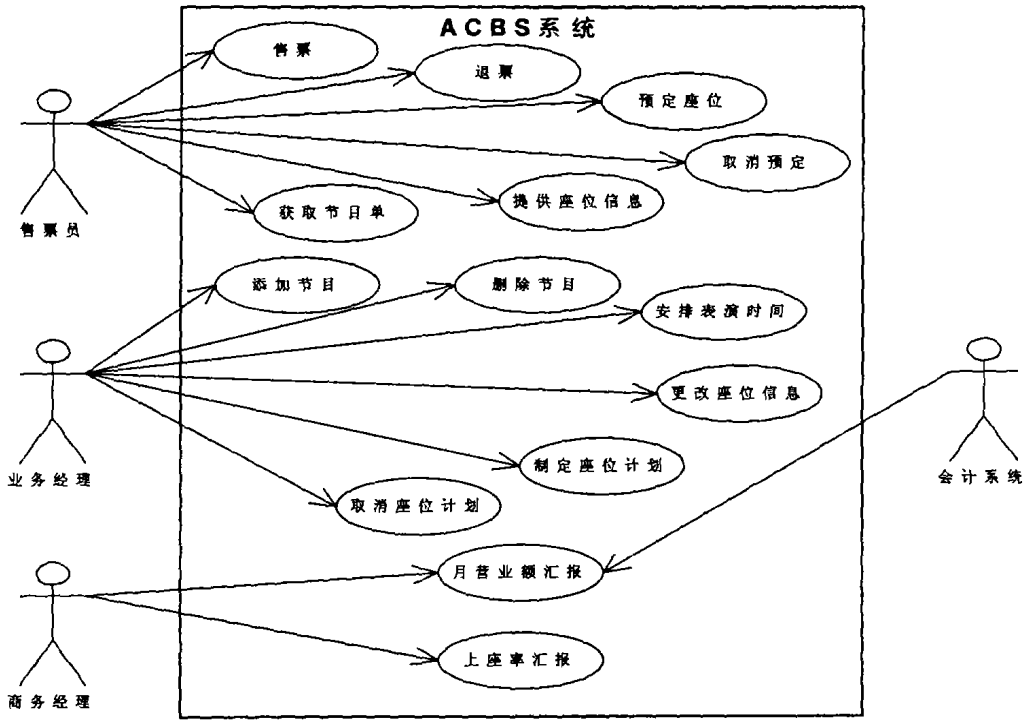

图3 艺术中心订票系统(ACBS)用例图

结束语 用例在软件需求的分析与获取中有其独有的优越性。本质用例作为用例概念的发展，在保留了这些优越性的同时，克服了传统用例所具有的局限性。相对于传统用例，本质用例更为简洁抽象。由于不涉及任何实现细节，可以支持更快速的开发过程。

Constantine 和 Lockwood 提出本质用例的最初目的是为了支持用户界面设计。随着对本质用例认识的深入，人们开始尝试将其应用到其他领域。在研究中我们发现，本质用例可以用来支持更为普遍的软件需求分析与获取。

本文首先介绍了本质用例的相关概念，并结合实例说明了基于本质用例的需求分析与获取过程。在整个的软件系统开发过程中，需求的分析与获取无疑是相当关键的，而本质用例为我们提供了一种行之有效的手段。

参 考 文 献

1 Jacobson I，et al. Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley，1992

2 Constantine L L，Lockwood L A D. Software for Use: A Practical Guide to the Models and Methods of Usage Centered Design. Addison-Wesley，1999

3 Constantine L，Lockwood L. Structure and Style in Use Cases for User Interface Design，2000

4 Jacobson I，Booch G，Rumbaugh J. The Unified Software Development Process. Addison-Wesley，1999

5 Wirfs-Brock R J. Designing Scenarios: Making the Case for a Use Case Framework. The Smalltalk Report，3(3)，1993

6 Biddle R，Noble J，Tempero E. Patterns for Essential Use Cases: [Technical Report CS-TR-01/02]. 2000

7 Biddle R，Noble J，Tempero E. Essential Use Cases and Responsibility in Object-Oriented Development，2001

8 Oberg R，Probasco L，Ericsson M. Applying Requirements Management with Use Cases，Rational Software White Paper TP505，2000