CN43-1258/TN 计算机工程与科学 1999年第21卷第4期
ISSN 1007-130X COMPUTER ENGINEERING & SCIENCE Vol.21,No.4,1999

现代软件开发方法中的Use Case技术*

The Use Case Technology in Modern Software Development Methods

刘作伟 宁 洪
Liu Zuowei *and* Ning Hong
(国防科技大学计算机学院)
(School of Computer Science, National University of Defense Technology)

摘 要 现代软件开发方法普遍采用了Use Case驱动的方法。本文依据UML 1.1规范，首先介绍Use Case的有关概念，然后介绍分析、设计和实现Use Case的过程，重点阐述Use Case的描述、细化和实现的有关细节，最后强调了分析、设计和实现Use Case过程中应该注意的一些问题。

ABSTRACT The Use Case-driven approach is used widely in most software development methods. According to UML 1.1, this paper introduces some concepts about "Use Case", describes the analysis, design and realization of Use Case processes, and details its description, refinement and realization. Finally, some issues that should be noticed in these processes are emphasized.

关键词 Use Case，使用实体，软件模型，对象。
KEY WORDS Use Case, actor, software model, object.

1 引言

在面向对象的软件开发方法中，客户关心的是软件系统的功能，而开发工作是围绕目标软件系统中的对象模型来进行的。为了把对象模型和功能模型有机地结合起来，在众多的面向对象开发方法中都采用了一种称为Use Case的技术。Use Case的分析、设计和实现与面向对象的分析、设计和实现结合起来，提供了一种贯穿整个软件生命周期的开发方式，使得软件开发的各个阶段的工作自然、一致地协调起来。在UML系统建模规范中，更把Use Case作为一个重要组成部分。下面以UML中的相关模型为例，介绍Use Case技术。

* 收稿日期：1998年11月11日；本课题受CCX基金资助。
作者简介：刘作伟，男，1973年5月生，硕士生，主要研究方向为面向对象技术、软件工程；宁洪，女，副教授，主要研究方向为数据库、软件工程。
通讯地址：410073 湖南长沙国防科技大学计算机学院
Address: School of Computer Science, Nat'l Univ. of Defense Tech., Changsha, Hunan 410073, P. R. China

2 Use Case 及其相关概念

2.1 定义

1995 年，Jacobson 把 Use Case 定义为：一个 Use Case 描述在一个系统中一组事务(transaction)执行的先后序列，这组事务的执行将向与该系统交互的使用实体(actor)返回可以度量的结果。此定义涉及到以下几个概念：

• 一个 Use Case：描述了系统中要发生的一个事件流，其中包括了具体的事件和事件发生的先后次序。

• 使用实体：在使用系统功能时，某个人或事物所充当的角色。同一个人或事物可以充当多个使用实体角色。一个 Use Case 可以与多个使用实体进行交互。使用实体可以是类、系统、子系统、另一个 Use Case 等。

• 事务：系统为完成某个功能而要执行的操作、事务具有原子性。

• 可以度量的结果：意味着事务序列的执行所返回的结果是可以预见或可以计量的，对使用实体来说是可以预计和有意义的。

一个系统中所有 Use Case 定义的集合就规定了该系统的全部功能。

2.2 Use Case 之间的相互关系

Use Case 之间存在两种关系，即扩展关系和使用关系。

(1) 扩展关系(extend)

A、B 是不同的 Use Case，A 扩展 B 的含义是，B 的一个执行过程可以引发 A 中定义的行为（在 B 的一个扩展点上且扩展点条件为真时）。一个 Use Case 可以被多个 Use Case 扩展（有多个扩展点）。Use Case 之间的扩展关系对应着 Use Case 的细化过程，这个细化过程具有两个特点：①是增量式的，即当系统中已经存在一个解决某个问题的具体途径的定义时，为解决另外的问题可以把更多的系统行为加入到该定义中，从而扩充、细化系统的功能；②这种扩充、细化工作并不直接修改系统中已经存在的定义。

(2) 使用关系(use)

A 使用 B 的含义是，A 可以使用 B 定义的行为，即 A 的执行过程中必定包括 B 中定义的行为。一个 Use Case 可以使用多个 Use Case。使用有两层意思：使用 Use Case 的结果（强调被使用 Use Case 所返回的结果）或使用 Use Case 中所定义的行为（强调 Use Case 的执行对系统产生的影响）。使用关系反映了共享的概念，即多个 Use Case 共享一个 Use Case 中定义的行为。

2.3 Use Case 类型和 Use Case 实例

Use Case 实例刻画了目标系统为完成某个功能而要执行的一个具体的动作序列，即一个事件流。如果目标系统中存在多个事件流、并且这些事件流具有相似性（包含的事件和事件发生的先后次序相似）时，则可以定义一个 Use Case 类型，用此 Use Case 类型实例化一次即得到一个 Use Case 实例。因此，标识和定义 Use Case，实际上就是要定义 Use Case 类型。

3 Use Case 的分析、设计和实现

3.1 Use Case 的分析和描述

3.1.1 识别、定义和描述 Use Case

先识别出 Use Case，然后定义相关的使用实体，再进行 Use Case 的描述工作。Use Case 的描述工作从开发者的角度来文档化 Use Case 的内容。Use Case 的描述至少应包含下面的内容：use case name(Use Case 的名称)、actors(使用实体)、participants(包含的参与者，如目标软件系统中的对象)、parameters(外部实体与 Use Case 相互交换的参数)、pre(在 Use Case 实例执行前系统应该满足的条件)、post(在 Use Case 实例执行后系统应该满足的条件)、relationships(Use Case 参与的关系的描述)。

3.1.2 Use Case 的精化和具体化

精化的本质就是给软件模型中的每个模型元素（如子系统、类等）收集所有的 Use Case。精化一个 Use Case 可以得到多个子 Use Case，子 Use Case 之间相互协作完成被精化的 Use Case 所定义的功能。子 Use Case 之间以“使用实体/Use Case”（其中一个 Use Case 作为另一个 Use Case 的使用实例）的关系相互联结，被精化的 Use Case 的使用实体一定是一个或多个子 Use Case 的使用实体。

具体化工作是指按照“类型/实例”的方式来刻画 Use Case 的内容。一个 Use Case 可以描述多个执行场景，一个执行场景对应着一个 Use Case 实例。因此，Use Case 实例化是 Use Case 具体化的方式之一。这种具体化过程中所做的工作有：Use Case 中包含的所有可能的执行场景的描述；Use Case 中包含的分支条件的确定等。

精化是用 Use Case 来精化 Use Case，具体化是用所包含的执行实例来刻画 Use Case 所包含的内容。在软件开发过程中，应通过有效地使用 Use Case 的精化和具体化手段将 Use Case 的描述、设计和实现有效地联系起来。

3.1.3 组织和管理 Use Case

随着 Use Case 分析工作的不断进行，需要采用一种方式来有效地组织和管理 Use Case 模型。在 UML 规范中，以包(package)的嵌套层次结构的形式来组织 Use Case 图。每个包中至少包含一个 Use Case 图，作为包中包含的 Use Case 的上下文图。在上下文图中，把 Use Case 的描述同软件动态设计模型中的相关部分联系起来，就可以有效地管理 Use Case 的分析、设计和实现过程。

3.2 Use Case 的设计和实现

在 Use Case 的分析工作之后，要采用相应的方式来实现其中定义的功能。下面介绍两种设计和实现方式，采用 UML 规范中的两种动态模型：协作图和状态图。

3.2.1 协作图

协作图中包含了对象、对象之间的消息传递，若其中包含有子系统，则表明该协作图还包含子协作图。消息分为同步和异步两种类型，而且对象之间的协作可以是并发的。软件模型中协作图的集合构成了协作模型。协作模型也表明了对象之间的关系，这种对象之间的静态关系是通过动态建模来体现的。

一个协作图描述了软件模型中的模型元素相互协作来实现一个 Use Case，完成系统的一个功能的过程。协作过程就是消息的传递过程，其中包含了使用实体向 Use Case 发送的消息。一个消息用协作图中一个对象方法成员来实现。

3.2.2 状态图

状态图包含了一组状态和一组状态转移。目标系统中每个实体都可以用状态图来刻画实体在生命周期内的行为。当 Use Case 中只包含一个软件实体(类)时，就适合用状态图来实现它。

使用实体向 Use Case 发送消息，就会产生一个调用事件(callevent)，从而引发一个状态转换，执行相应的动作；还可以引发新的事件，进而引发新的状态转换等等。其中，使用实体还可以引发新的调用事件。

上述两种方式的一个关键区别是如何对待对象之间的请求。协作图采用操作，状态图采用信号。在 UML 中操作和信号均是请求的子类型，详细内容请参阅 UML 规范。当 Use Case 涉及一个软件实体（典型情况下，该软件实体本身比较复杂）时，则采用状态图方式；而当 Use Case 涉及多个软件实体，并且要反映出各个软件实体之间的关系时，就适合采用协作图方式。另外，也可以采用其它的方式来设计和实现 Use Case。例如，当要强调事件（或操作）的先后顺序，强调时间约束条件时，应当采用 UML 中的顺序图方式。有关其它的设计和实现 Use Case 的方式可参阅文献 [2～4]。

4 结束语

· 在面向对象的软件开发方法中应用 Use Case 应尽量遵从以对象为中心的特点，把 Use Case 技术有机地集成到具体的面向对象开发方法中去。

· 在软件生命周期中采用 Use Case，应该规范 Use Case 的定义、描述和设计。

· 维持系统分析和系统设计之间合理的、自然的分离。在分析阶段，Use Case 主要解决“做什么”的问题；而在设计阶段，主要解决“如何去做”的问题。

· 在面向对象的软件开发方法中，Use Case 的分析和设计可以作为对象外部接口的分析和设计的依据，从而有助于信息隐藏。

Use Case 作为软件开发过程中的一种强有力的辅助手段，为问题的提出、理解和解决提供了一个规范的途径。在开发过程中，应当把 Use Case 技术有机地集成到软件开发方法中去，并在开发的各个阶段注意保持开发方法的特点和内在统一性。

如何在实际的开发过程中更好地进行 Use Case 的分析、设计和实现工作，还需要做进一步的研究。各种软件工具（特别是分析、设计和软件测试工具）要为应用 Use Case 技术提供有力的支持。

参 考 文 献

1 Berard E. Be Careful with Use Cases. 1998.5. http://www.toa.com/pub/html/use_case.html

2 UML Semantics. Version 1.1. Rational Corp.1997

3 UML Notation Guide. Version 1.1. Rational Corp.1977

4 Hurlbut R. The Three R's of Use Case Formalisms: Realization, Refinement, and Reification. Technical Report: XPT-TR-97-06. Expertech Ltd.1997