第27卷 第3期 桂 林 工 学 院 学 报 Vol. 27 No. 3
2007年8月 Journal of Guilin University of Technology Aug. 2007

文章编号：1006－544X（2007）03－0442－04

基于树结构的用例分层扩展

王 挺[a]，麦范金[b]，程小辉[a]

（桂林工学院 a. 电子与计算机系，b. 现代教育技术中心，广西 桂林 541004）

摘 要：粗粒度的用例建模不适应大型软件系统的开发，通过引入树结构，对用例进行分层分解扩展，使用例细化. 但UML已定义的用例关系对于描述细化用例和原用例之间的关系并不适用，因此提出了一种新的用例间的关系，使细化后的用例与原用例之间能够继续保持紧密的结构关系，从而适应大型软件系统的开发.

关键词：统一建模语言；用例；分层扩展；树结构

中图分类号：TP311.52 **文献标志码**：A

统一建模语言（unified modeling language，UML）是一种面向对象的建模语言，现在已成为软件建模的一个标准. UML具有扩展性，允许用户对其进行自定义扩展，以满足更广泛、更深层的建模需要.

用例（use case）是UML用户视图的重要组成部分，是一种用于分析系统功能和获得客户需求的方法. 用例能够清楚地表示一个系统、一个模块甚至一个类的功能[1]. 在面向对象的系统开发方法中，如何更好地细化用户需求、精化系统功能已成为广大系统分析员的目标. 用例的细化成为系统建模过程中非常重要的一环.

然而，至今所有的UML版本都没有提供一种确切的细粒度的用例分析与设计的方法. 在UML中，描述用例关系一般采用包含（include）和扩展（extend）两种方法. 包含主要用于描述用例间的功能调用关系，扩展则主要用于描述用例的向外功能扩展[2]. 而对于用例的内部功能分割和用例细化，UML则没有明确的表述方法[3]. 当UML需要对一个粗粒度用例详细描述时，往往只能使用多张结构松散的用例图进行分割描述. 这些松散的用例图无法体现一个分割后的用例与原用例之间的关系，无法使客户和系统开发人员对整个软件系统形成系统化、结构化的概念. 对于要精化到类和方法的后继开发工作（如详细设计）来说，这种系统建模显然不适用. 对于大型项目的开发来说，就更加不适合了. 本文提出一种基于树结构的用例分层扩展，并由此建立一种新的用例间的关系——分解（divide）关系，用于说明用例的内部功能分割和用例细化，并通过实例说明分解关系能够很好地实现用例图之间的结构化和系统分析的细粒度，从而有效解决上述问题.

1 UML用例分层扩展

1.1 树结构与分层

树是数据结构中的一个重要概念，主要用于描述非线性数据结构. 树以其分支关系定义其层次结构. 根为第1层，根的孩子为第2层. 某节点在第l层，则其子树的根就在第$l+1$层. 树中节点的最大层次称为树的高度.

1.2 UML用例分层

UML 2.0使用主体（subject）的概念替换了以前版本中系统（system）的概念[4]. 但不论主体还是系统，它们都可以被看作是用例的集合，反映了系统功能和客户需求. UML对其用例并没有具体的粒度划分规则. 但一般来说，在系统建模初

收稿日期：2006－05－09
基金项目：广西自然科学基金资助项目（桂科自0447099）
作者简介：王 挺（1981－），男，硕士研究生，研究方向：人工智能、软件工程.
通讯作者：麦范金，副教授，E-mail：mfj@glite.edu.cn.

期，首先定义 1 个或多个粗粒度的用例，然后根据客户需求，不断将粗粒度用例进行功能切割，使其精化，同时达到用例粒度细化的目的. 假设每个软件系统都只有 1 个原始的用例，即用例全集，用 I 来表示. 所有在建模过程中第 1 次用例划分所得的粗粒度用例 ($I.1, I.2, \cdots, I.n$)，都是 I 的子集，即

$$I = \bigcup_{i=1}^{n} I.i.$$

在第 1 次用例划分后，把所得的每个粗粒度用例都扩展为原系统的一个子系统，并对其进行精化，即将其分割成更细粒度的用例，则第 2 次用例划分后所得的用例是第 1 次的子用例. 例如，假设 $I.a$ 是 I 经过第 1 次用例划分后所得的 1 个粗粒度用例，即 $I.a$ 是 I 的 1 个子集. 则在第 2 次划分用例的时候，首先把 $I.a$ 扩展为 I 的 1 个子系统，然后将 $I.a$ 细化成更细的用例($I.a.1, I.a.2, \cdots, I.a.n$). 同理，有

$$I.a = \bigcup_{i=1}^{n} I.a.i.$$

这时，$I, I.a(1 \leqslant a \leqslant n)$ 与 $I.a.b(1 \leqslant a \leqslant n, 1 \leqslant b \leqslant n)$ 就构成了一个 3 层结构. 当然 $I.a.b$ 可以被继续划分成更细的用例，同时整个系统用例也构成了一棵以 I 为根的树(图 1). 可以看到，子用例是由父用例分解细化得到的. 父用例和子用例之间存在功能继承的关系，这种关系可定义为用例间的一种新关系 —— 分解(divide) 关系.

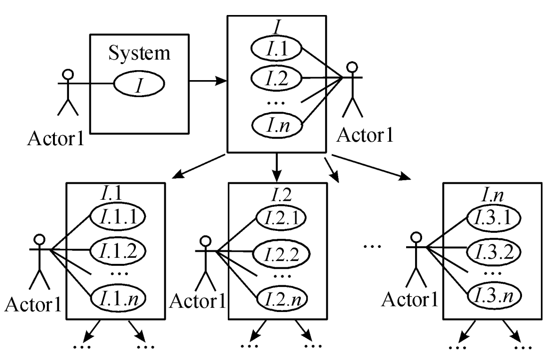

图 1　用例树示意图
Fig. 1　Use case tree

2　一种新的用例关系

2.1　分解关系

UML 的用例之间存在两种已定义的关系，即包含和扩展. 其中包含关系也可称为使用（uses）关系[5]. 本文所提出的分解关系是用例间的第 3 种关系. 如上所述，分解关系一般是用来描述父用例和子用例这类粒度大小不同的用例间的关系. 通过用例分层扩展和分解，下层用例所描述的客户需求和系统功能将是上层用例的精化和细化.

与包含和扩展采用箭头来描述用例间关系不同，分解关系采用具体的命名规则来体现用例间的关系，即在用例名称前加上用例编号. 所有用例均采用用例编号加用例名称的方式进行命名. 可以这样认为：分解关系是一种通过字符编号来说明的一种用例间的关系.

给用例编号可以采用与上一节的假设相类似的方法来进行. 这里引入 C + + 中的圆点操作符(.). 凭借圆点操作符，既可以将用例名称限制在某个名字空间内，又可以将用例的粒度细化到类和方法的层次. 这样符合详细设计的要求. 圆点操作符可以在同一用例编号中多次使用. 圆点操作符前的字符表示父用例在上层的号码，圆点操作符后的字符表示用例在当前层的号码. 假设某用例编号中圆点操作符的个数为 n，则该用例在其用例树中的深度为 $n+1$. 给用例编号，能够清晰地反映逻辑上的用例树的结构. 本文在第 3 节给出 1 个公司人资管理系统的 UML 用例建模的实例来进一步说明用例分层扩展和分解关系.

2.2　层与层之间的特点

采用分解关系对系统用例分层之后，层与层之间具有以下特点：

（1）下层用例反映上层用例的功能，上层用例的功能为下层所有用例功能的整体抽象.

（2）上层用例的接口（interface）为该用例下层所有用例接口的并集.

假设：A 为上层用例，$A.1, A.2, \cdots, A.n$ 为 A 下层子用例，且 $A.1 \cup A.2 \cup \cdots \cup A.n \subseteq A$，

则
$$\text{Interface } A = \bigcup_{i=1}^{n} \text{Interface } A.i.$$

(3) 上层用例的角色为该用例下层所有用例角色的并集. 即：

$$\text{Actor } A = \bigcup_{i=1}^{n} \text{Actor } A.i.$$

2.3　包含、扩展与分解三者间的关系

包含主要用于描述用例间的调用关系. 比如用例 A 需要功能 b，而功能 b 已在用例 B 中定义. 则此时 A 只需调用 B 即可，而不用再重新定义一遍 b. A 和 B 之间不是隶属关系.

扩展用于描述用例的非必需的子功能. 比如用例 B 具有功能 b,但 b 对用例 A 来说不是必需的功能. 用例 A 在没有 b 的情况下依然完整可用,但用例 B 的加入能使用例 A 的功能得到扩展和提升,则 A 与 B 间就可视为扩展关系. 这种关系是可有可无的隶属关系,并且 B 可能同时隶属多个用例,为多个其他用例的共享扩展.

分解用于描述用例的必需的子功能. 比如用例 A 必须要有用例 B 提供的功能 b 才能完整执行功能 a,则 B 对 A 来说是必需的. B 是 A 的一个必需的子用例. 这样,A 与 B 之间就可视为具有分解关系. 用例 A 被分解成了若干部分,其中一部分就是 B. 分解关系是一种必要的、非可有可无的隶属关系. 用例 B 只隶属于用例 A,且必定在用例 A 的下层.

3 实 例

现在各公司都把人力资源管理列作公司核心管理的一部分[6]. 人力资源管理的重要性,使得人资信息系统具有了广阔的市场前景. 但一般来说,由于人资系统应用面较广,所以模块众多,架构也比较庞大. 如果开发者在系统分析设计的时候不能理清系统结构,则不利于整个项目的成功完成. 这里以一个人力资源管理系统为例,对其用例分层扩展,并运用分解关系来说明层与层之间的用例关系.

图 2 描述了人力资源管理系统的用例全集,即系统用例树的根用例. 对其进行第 1 次用例划分后,得到如图 3 所示的用例图. 可以看到整个人资系统被划分成六大主要功能块. 由于系统较大且篇幅有限,为简要说明问题,本文仅选择其中的部分用例来进一步分层扩展. 现在对 $I.1$ 划分后,得到如图 4 所示的用例图. 选择 $I.1.1$ 进行下一步细化,得到如图 5 的用例图. 再选择 $I.1.1.1$ 进行下一步细化,可以得到如图 6 的用例图. 可以清楚地看到,图5已精确到类的层次,而图6

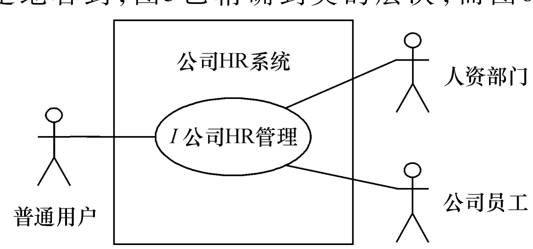

图 2 *I* 公司 HR 系统

Fig. 2 HR System of *I* company

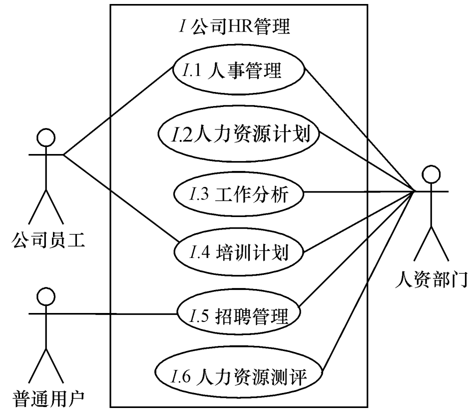

图 3 *I* 公司 HR 管理

Fig. 3 HR management of *I* company

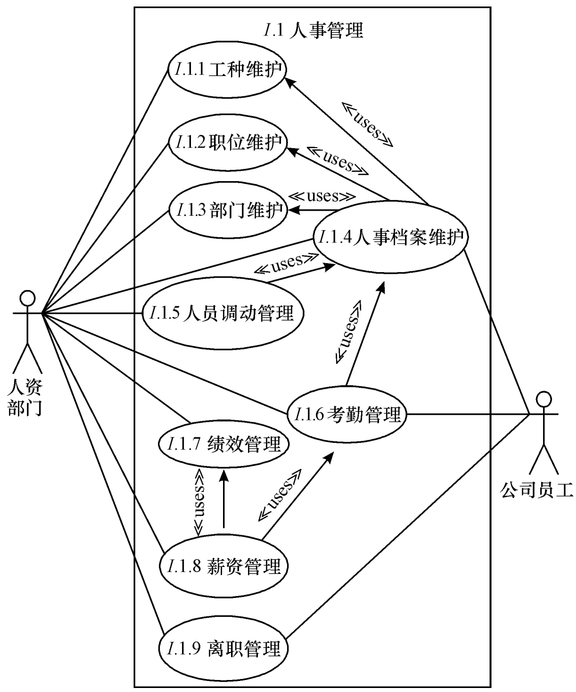

图 4 *I*. 1 人事管理

Fig. 4 *I*. 1 personal management

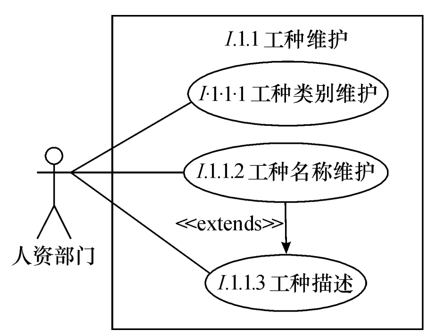

图 5 *I*. 1. 1 工种维护

Fig. 5 Work maintenance of *I*. 1. 1

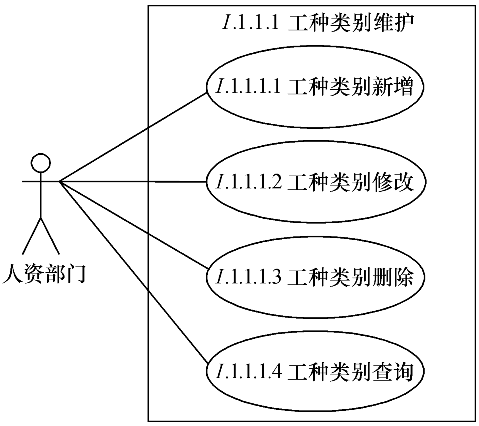

图 6　I. 1. 1. 1 工种类别维护
Fig. 6　Work types maintenance of I. 1. 1. 1

更是将类的方法描绘了出来. 整个用例分层扩展一共进行了 5 层. 图 7 描述了用例分层扩展得到的用例树. 通过用例树，可以清楚地看到，分解关系既可以实现用例细化，又可以实现细化用例与原用例之间的结构化.

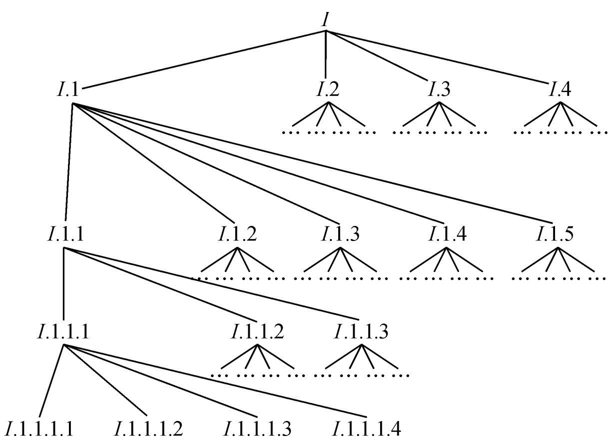

图 7　人力资源管理系统用例树（部分）
Fig. 7　Use cases tree of HR system

4　结　论

粗粒度的用例建模不适合大型项目的开发，UML 也没有提供具体的关于用例细化的标准和解决方案，且 UML 现有的用例关系也无法表述用例细化之后，细化用例和原用例之间的关系. 运用本文所提出的基于树结构的用例分层扩展及用例间的分解关系可以很好地实现用例细化，以及描述用例细化之后的结构关系. 同时，对于参与大型项目的开发人员来说，用例分层扩展和分解关系可以缩短其了解系统的整体架构和系统各模块功能的时间，从而提高软件系统开发的效率.

参考文献：

[1] Albir S S. UML 技术手册［K］. 北京：中国电力出版社，2002：185－192.

[2] Pender Tom. UML 宝典［M］. 北京：电子工业出版社，2004：287－316.

[3] 陈昊，应时，景奕昕，等. 一种用目标驱动技术扩充的用例分析方法［J］. 计算机工程与应用，2004，40（6）：66－68.

[4] Object Management Group. Unified Modeling Language：Superstructure［EB/OL］.（2005－7－4）.〔2006－5－9〕. http：//www. omg. org/cgi-bin/apps/doc？formal/05－07－04. pdf：569－586.

[5] Whitten J L，Bentley L D，Dittman K C. 系统分析与设计方法［M］. 北京：机械工业出版社，2004：233－250.

[6] 罗鸿，王忠民. ERP 原理 · 设计 · 实施［M］. 北京：电子工业出版社，2003：288－292.

Layer Expansion of Use Case Based on Tree

WANG Ting[1]，MAI Fan-jin[2]，CHENG Xiao-hui[1]

（1. *Department of Electronics and Computer Science*，*Guilin University of Technology*，*Guilin* 541004，*China*；
2. *Modern Education Technology Center*，*Guilin University of Technology*，*Guilin* 541004，*China*）

Abstract：Coarsely granular use case modeling can′t meet the development of large scale software system. UML can not provide a proper detailed method on use case. A tree-based layered expansion of use cases analyses is introduced，which can help use cases thinner and more detailed. The relationships between use cases，such as “include” or “extend”，are not able to illustrate the new relationship between use case and the new one. Therefore，a new relationship between use cases is put forward to make use cases more detailed.

Key words：UML；use case；layer expansion；tree structure