

# 用例驱动的软件体系结构构建

刘韶涛 余金山

(华侨大学信息科学与工程学院, 福建 泉州 362011)

**摘要** 软件体系结构是有效实现大粒度软件复用的重要方法, 但如何具体实现软件体系结构一直没有得到有效的解决. 文中在分析用例和软件体系结构之间关系的基础上, 提出一种用例驱动、迭代增量方式, 构建软件体系结构的方法及其模型. 同时, 分析增量迭代的过程和相关问题.

**关键词** 软件体系结构, 软件复用, 用例, 迭代增量

**中图分类号** TP 311.5

**文献标识码** A

软件复用技术可以极大地提高软件生产率、降低软件成本、增强软件产品的可靠性, 它是克服软件危机的行之有效的办法. 近年来关于软件复用的研究, 人们主要集中于功能部件的复用上, 这也是源代码级的复用. 然而, 随着用户需求的提高和软件技术的快速发展, 在软件的设计开发过程中, 特别是对大型的、复杂的软件系统而言, 开发者不仅要考虑到功能部件的设计和复用, 而且更要考虑到软件体系结构的设计和复用问题<sup>[0,2]</sup>. 在软件工程领域中, 软件体系结构(SA)已经成为越来越重要的课题, 是设计的技术基础. 它可以作为满足需求的框架, 促使软件工程的重点从功能向结构转移, 以支持大型的复杂的软件系统的开发和维护. 但是, 整个软件体系结构技术从理论到实践还处在发展的时期<sup>[6]</sup>, 是抽象意义上的概念模型, 如何构建软件体系结构并没有切实可行的办法. 本文就如何解决构建软件体系结构的问题, 提出了一种用例驱动、迭代增量的构建方法, 并就相关的问题进行了分析探讨.

## 1 软件体系结构的概念

软件体系结构关心和描述的, 是大型软件系统在组织结构特性方面的性质. 在很多大型软件系统开发的各种活动中, 人们发现软件体系结构的正确设计和选择, 往往是整个软件系统最终成功的最为关键的因素. 如果在系统分析和设计阶段, 系统工作人员没有得到或选择正确的系统软件体系结构, 则会给将来的整个开发活动造成灾难性的后果<sup>[6]</sup>. 软件体系结构包括了对以下4方面所作的决策: (1) 软件系统的组织; (2) 构成系统的结构元素和各元素之间的接口, 以及由元素间各种协作所规定的各元素行为; (3) 结构元素和行为元素合成为逐渐增大的子系统; (4) 指导这种组织的软件体系结构风格、元素, 以及它们的接口、协作和组合. 但是, 软件

体系结构不只涉及结构和行为, 还涉及到使用、功能、柔性、重用、可理解性、经济性和技术约束, 以及折衷方案、美学等. 一个大型的、复杂的软件系统从某些方面来讲, 它通常是无先例可循的, 或是独一无二的. 它常采用未经证实的技术或各种技术的新颖组合, 并把现有的技术推向极至, 且在构造时还必须适应将来一系列的巨大变化. 随着系统的日益复杂, 设计问题超过了计算的算法和数据结构, 设计和确定整个系统的结构便成了一类新的问题<sup>[4]</sup>.

## 2 用例和软件体系结构的关系

用例(Use Case)是能够向用户(不仅仅是人, 也可以是其它系统) 提供有价值结果的系统中的一种功能. 用例获取的是系统功能需求. 所有的用例合在一起构成用例模型, 它描述了系统的全部功能. 该模型代替了传统的系统功能说明. 用例模型所涉及的系统细节最少, 而离系统需求最近. 用例模型的主要目的是定义系统要完成什么功能, 从而使用户和软件工程师达成一致. 它用于驱动软件系统的其它开发工作, 也是对象建模活动的起点. 用例模型由角色类型和用例类型组成. 图 1 为某 ATM 系统用例模型的简单例子, 它由一个角色和几个用例组成. 后面的订单处理系统实例也给出了用例的例子.

用例和软件体系结构之间存在着紧密的关系. 一方面, 软件体系结构受到我们所希望系统支持的用例的影响, 用例驱动软件体系结构的建立. 软件体系结构受到以下因素的影响, 如图 2 所示. 可以把图 2 中右侧的几项看作是, 能使我们以某种方式来建立软件体系结构的约束和

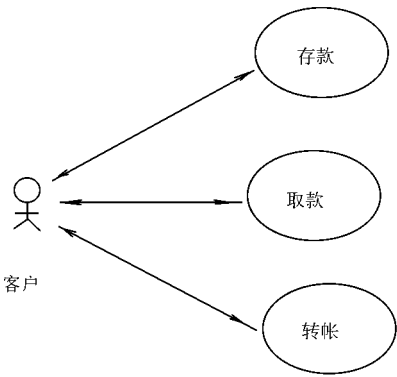


图 1 ATM 系统用例模型

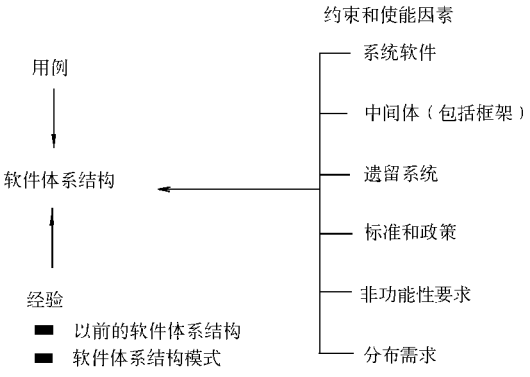


图 2 影响软件结构的因素

使能因素. 另一方面, 用例也受到软件体系结构的影响. 软件体系结构是在细化阶段的迭代过程中被创建的. 开始先确定软件体系结构的高层设计, 例如一个分层体系结构. 然后在第 1 次迭代的几次构造中逐步确立该体系结构. 在细化阶段的最后, 我们得到的是一个相对稳定的软件体系结构. 当建立了一个稳定的软件体系结构后, 就可以通过在构造阶段实现其余的用例来实现系统的全部功能. 在开发构造阶段中实现的用例, 主要是以客户和用户需求作为输入. 但这些用例也会受到细化阶段所确定的体系结构的影响, 如图 3 所示.

在捕获新的用例时, 可以利用对已经存在的体系结构的知识来更好地完成我们的工作. 在评估每个所选用例的价值和成本时, 也是根据现存的体系结构进行的. 例如, 客户希望获得一种监控处理器负载的功能. 可以将这个要求说明为测量计算机高优先级负载的用例. 要实现这个用例, 需要对现在所用的实时操作系统进行一些改动. 而开发组则建议, 通过一个独立的外

部设备呼叫系统并测量应答时间, 来实现所需的功能. 于是, 客户得到了一种更为可靠的方法, 而开发组也可避免对关键的底层体系结构进行改动. 我们可以与客户进行协商, 以确定是否可以改变用例, 以使用例和设计结果与已有的体系结构更加一致, 从而使实现更为简化. 通过使用例与体系结构保持一致, 便可以利用现存的资源有效地创建新的用例、子系统和类.

所以, 一方面软件体系结构受到我们希望系统支持的用例的影响, 用例驱动软件体系结构. 另一方面, 当我们将需求捕获为用例时, 可以利用对体系结构的知识来更好地完成任务. 软件体系结构指导用例, 如图 4 所示.

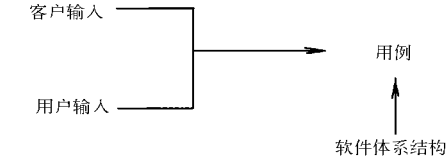


图 3 影响用例的因素

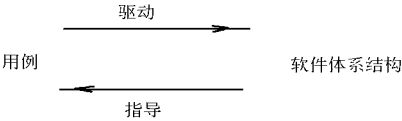


图 4 用例与软件结构的关系

3 用例驱动的软件体系结构迭代增量构建过程

根据用例和软件体系结构的相互关系, 我们可以通过迭代增量的方式来构建软件体系结构, 如图 5 所示. 首先在很好地了解领域范围的基础上, 建立一个临时的软件体系结构, 但不考虑具体的用例. 接着选取几个重要的用例, 并进一步使体系结构能够支持这些用例. 然后再选

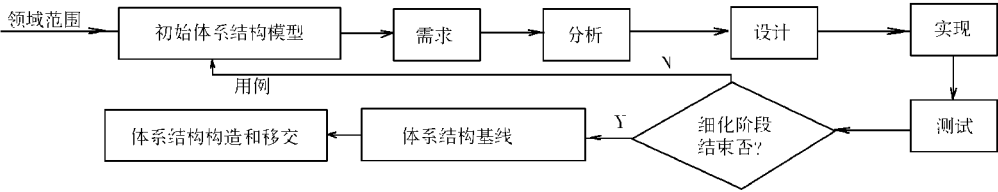


图 5 用例驱动的软件结构迭代增量构建模型

取更多的用例, 建立更加完美的体系结构. 依次类推. 在每一次迭代中, 都选取并实现一组用例来确认体系结构. 如果必要, 再对体系结构进行改进. 随着每次迭代的进行, 我们还在所选用例的基础上, 进一步实现体系结构的专门应用部分. 因此, 在通过迭代实现整个系统时, 用例有助于我们逐渐完善体系结构.

一个软件开发项目, 可粗略分成初始和细化阶段与构造和移交阶段两大块. 初始阶段的主要目标是设定产品应该做什么的范围, 并建立初始业务案例, 从业务的角度表明项目的可行性. 细化阶段的主要目标是建立体系结构基线, 捕获大多数需求. 构造阶段的主要目标是开发整个系统, 并确保产品可以开始移交给客户, 即产品达到最初的可操作能力. 移交阶段的主要目标是, 确保得到一个准备向用户社团发布的产品. 在这个阶段, 需要培训用户如何使用该软件. 软件体系结构主要是在细化阶段的迭代过程中发展起来的. 每次迭代的进行都要经历 5 种核心工作流, 即需求、分析、设计、实现和测试. 细化阶段的最终结果是一条体系结构基线——一个只有很少软件“肌肉”的系统骨架. 在细化阶段最后, 我们从体系结构的角度开发了代表最重要的用例, 以及其实现的系统模型. 我们也确定了要遵守哪些标准, 要使用哪些系统软件和中间件, 要重用哪些遗留系统, 以及有哪些分布需要. 这样, 我们就获得了用例模型、分析模型、

设计模型, 以及其它模型的早期版本. 这些模型的集合就是体系结构基线.

虽然每次迭代都要经历需求、分析、设计、实现和测试工作流, 但不同阶段的迭代侧重点不同. 在初始和细化阶段中, 绝大部分工作集中在捕获需求, 进行初步的分析与设计上. 在构造阶段中, 重点则转移到详细设计、实现和测试上. 每次迭代在其结束时进行评估. 其中的一个目的就是判定是否提出了新的需求, 或者现有需求的变化是否影响到后续的迭代. 一次迭代产生一个增量结果. 一个增量是一次迭代的内部版本与下一次迭代的内部版本之间的差别.

迭代增量式体系结构构建向软件组织的工作实践提出了挑战. 它要求改变工作态度. 开发组织的侧重点不得不从统计代码行转移到降低风险, 以及建立体系结构功能基线上来. 运用迭代、增量的方法要注意以下 4 个重要结论. (1) 为了在初始阶段获得业务案例, 开发组织应该侧重于缓解关键风险. (2) 为了降低成本、减少缺陷和缩短上市时间, 开发组织必须使用可重用的构件. (3) 为了避免交付延期、预算超支和产品质量低劣, 开发组织必须“首先抓住要害”. (4) 为了避免在交付时所构造的产品过时, 开发组织不能固执地认为不应作任何修改. 分阶段迭代的方法使开发过程能够进一步沿着开发踪迹来处理变更.

图6是我们采用这种方法开发的某公司订单处理系统的系统级用例, 图7是对应的由订

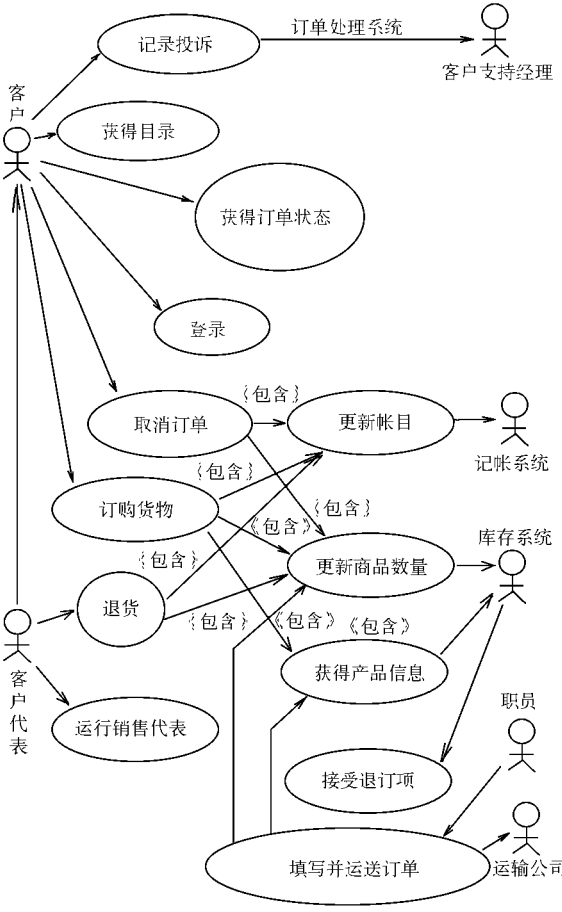


图 6 订单处理用例

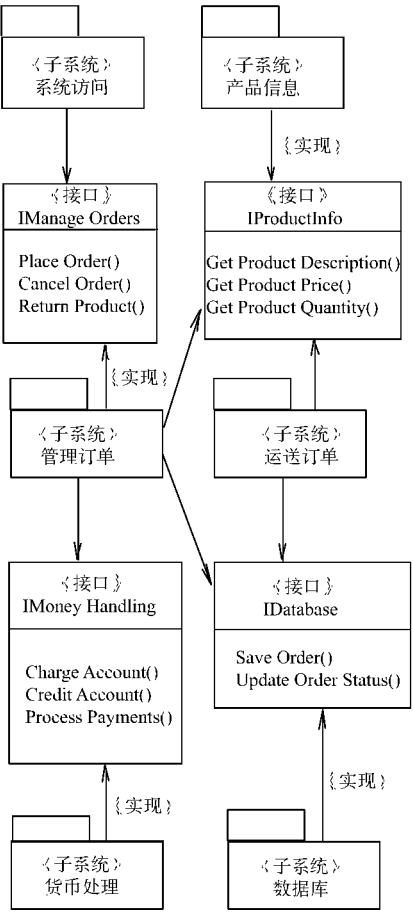


图 7 订单处理的体系结构图

单处理系统用例驱动的订单处理系统体系结构. 限于篇幅, 我们不再给出详细的迭代过程.

## 4 结束语

用例驱动、迭代增量的软件体系结构构建方法, 对于软件体系结构的构建具有很好的指导作用, 它符合人们的认识和思维方式. 实践证明, 这种方法相对于传统的体系结构构造方法相比, 具有更高的效率. 在迭代中引进四阶段法(初始、细化、构造和移交), 可以设计出更好的软件体系结构和更好的迭代控制过程. 但是, 这种迭代和增量的开发方法不仅需要一种新的管理项目的方式, 而且需要支持这种方法的工具. 另外, 迭代用例的选择、排序和相关性, 以及如何保证每次迭代的增量和最后收敛等问题, 都有待于进一步的研究.

## 参 考 文 献

- 1 Perry D E, Wolf A L. Foundations for the study of software architecture [J]. ACM SIGSOFT Software Engineering Notes, 1992, 17( 4): 40 ~ 52
- 2 余金山. 试论软件过程模型及其重要性[J]. 华侨大学学报(自然科学版), 1994, 15( 1): 112 ~ 116
- 3 万建成, 卢 雷. 软件体系结构的原理、组成与应用[ M ]. 北京: 科学出版社, 2002. 34 ~ 37
- 4 Shaw M, Garlan D. Software architecture: Perspectives on an emerging discipline[ M ]. Englewood Cliffs: Prentice Hall, 1996. 20 ~ 40

# The Construction of Software Architecture Driving by Use Case

Liu Shaotao      Yu Jinshan

( College of Info. Sci. & Eng. , Huaqiao Univ. , 362011, Quanzhou, China)

**Abstract** Software architecture is an important method for the effective realization of the reuse of large grain size software. But how to concretely realize software architecture remains to be effectively solved. Based on the analysis of the relation between use case and software architecture, the author presents a method of iterative increment driving by use case for constructing software architecture and its model; and analyses the process of incremental iteration and relevant issues.

**Keywords** software architecture, software reuse, use case, iterative increment