软件再工程及可复用性

Software Reengineering and Reusability

贾 洞 郝振明

(浙江师范大学计算机科学与工程学院 金华321004) (暨南大学计算机系 广州510632)

Abstract Reengineering offers an approach to migrate an aging systems towards an evolvable systems. Reengineering is an opportune moment to capture viable assets from existing programs and make them available for future reuse. In this paper, we propose to address reusability issues in the context of reengineering, and explore means for integrating reengineering efforts with software reuse. We also discuss the main phases of the reengineering for reusability, then comment on cost- benefit analysis issuse, outline a reusability framework for reengineering and other relevant research projects.

Keywords Object-oriented, Software reengineering, Reusability, Reverse engineering

1 引言

许多老的商用程序有其严重缺陷:(1)经过多年的发展后,程序变得越来越复杂,维护代价高,程序得不到可靠、及时的修改;(2)许多老的程序需要进行大量的扩充和重新设计以满足新的需求,但要做到功能完善非常困难;(3)新的技术能全面降低计算费用,更灵活使用存储数据和简化系统用户界面,但许多老的程序无法采用新技术而只能运行在过时的、低效率的平台上。通常,为使系统有更好的可维护性而对程序进行重新构建以及为适合新的计算机、数据库、操作系统和语言等而进行的转换,并不能解决已有程序中的所有问题,事实上,这只能在短期内延长系统的寿命。从长远来看,软件必须重新编写以充分利用新技术的优点并满足各方面的需要。然而,当今软件系统的规模变得越来越大,结构也越来越复杂,同时从头开始构建的大系统数量在急剧地减少,因而很多老的系统正在被逐步地利用。在这种情况下,软件再工程变得越来越重要,因为它提供了一条把老的系统转换为可演化系统的现实可行的途径,是一种可以改进人们对软件的理解和改进软件本身的活动[2]。

软件再工程中包括了大量的程序代码的重新设计,因而其代价昂贵,也有风险。为了改善软件再工程的投资-收益率,软件的复用能从根本上带来开发程序方法的突破,我们希望在再工程过程中,能尽可能地从老的程序中提取可复用特性。在设计过程中,必须从程序中提取设计信息(有时是需求信息),把程序分成几个部分,并以一种新的方式提取和综合程序,以满足新的软件需求。当然,所有这些都是和我们深刻理解程序的全局和局部的特性紧密相关的,因此,再工程实际上是在合适的时机从已有的程序中获取有价值的信息以利于将来的复用。软件再工程和可复用性两者是互相促进的,可望能解决软件危机。

2 可复用性的再工程:生命周期各阶段

图1表明了可复用性的再工程生命周期的主要阶段。在软件系统的再工程之前须为再工程选择和优化候选系统制定一系列的规划,规划的制定基于性能分析、软件系统的商用价值分析(包括投资-收益率分析)。

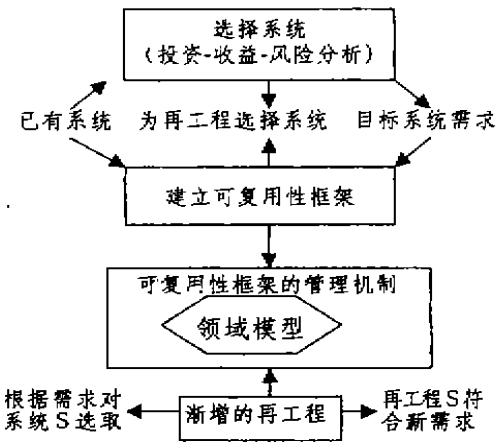

图1 可复用性的再工程生命周期图

在规划之后,选取系统是一个逐渐递增的再工程过程。在系统S的再工程过程中,通过可复用性框架的设计和代码的积累用以重构(re-construction)目标

系统S'，此期间，可复用特性的贮存库是随再工程各阶段而渐增的，系统S的新的恢复特征应加入未来复用的贮存库中。

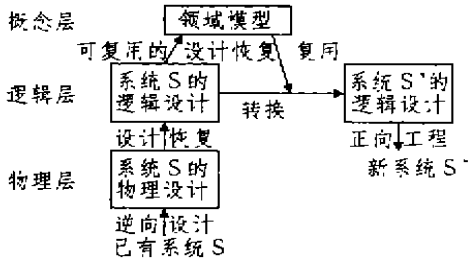

图2 再工程各阶段的逻辑结构

在再工程过程中，我们将一个已有的系统S转化为一个新系统S'以满足各种新的需求，图2为再工程阶段的逻辑结构，其中物理层可由逆向工程技术创建，包括控制流图、数据流图、过程调用树、数据结构图等抽象语法树和抽象设计；逻辑层包括ER数据模型和逻辑过程；概念层则提供了有关面向用户和领域概念的程序描述。各层次的映射使得从概念层到程序代码的信息跟踪变得更容易。在图的每个层次中，我们对程序视图进行了标识，以利于程序员对各个阶段的再工程和维护；同时，也为这些视图建模，为每个层次定义方法以了解和查询程序信息，并通过一些自动或半自动的技术进行设计恢复从低层次中获得深层次的程序抽象。这种抽象的层次比直接检查系统所得的抽象更高，它所产生的信息是提高软件构件可复用性的重要途径。

3 再工程规划

在为再工程选择候选系统时，一种方法是选择技术上有缺陷但仍有商业利用价值的已有系统，接着作进一步的分析，尽力估测一个软件再工程解决方法的投资-收益率，并从大量的估测中判别其成本、收益和风险，并以此作为选择候选系统的依据。

软件可复用性对再工程的投资-收益率有着直接的影响，可复用性框架必须在再工程设计前已经设计完毕。但这里有两个原因，可能会花费大量的时间。首先，程序的结构将被整个改变；其次，须从大量的潜在用户中抽取其特征，理清各部分间的实现的相关性和通用性。这样，一旦我们有了深刻的理解并经过合适的提炼，可复用的特性包括设计分片(design fragment)、数据模型、过程(procedure)等等，都将在新系统中得以充分应用。

4 再工程的可复用性结构

对再工程描述可复用性结构，这种结构为可复用设计和编码建立框架，同时也为分类、描述、存储和导出可复用特性等提供便利。这种可复用性框架包含两部分：领域模型(domain model)和可复用性的管理结构。我们可以通过给出一个系统的应用域，来对领域模型进行确定、分类和描述其潜在的可复用性特点，这些特点可能涉及对象、过程以及各种规范，在一个领域模型中，由结构表示的概念能帮助程序员理解一个给出的概念是如何实现的。两个并行的动作构成了领域模型：已有程序的逆过程和独立的领域分析。

可复用性的特点可通过面向对象建模加以体现，对象代表了从应用领域中提取的有意义的概念。在商用程序中，许多对象候选者被自然地从数据模型中选取出，对象包括数据模型和过程，是与特定的数据组相关的。具有可复用性的建模是从基于已有程序的数据结构分析和数据库规划的反复提炼中得来的，逆向工程的数据模型是与目标系统的新的需求和数据分析过程的进一步提炼相一致的；再则，对象模型是通过与数据实体相联系的过程分析得来的，对象间的关系可导出实体关系，并可通过分析相互联系的对象导出更多的对象关系。除此之外，还需对规则、事件、控制等模型化。

在此，必须强调建立一个面向对象的领域模型对于处理可复用特性的重要性。如果目标是获得面向对象的程序结构，那么建立一个面向对象的领域模型将非常有效。事实上，对象模型的主要目的是组织程序信息便于理解和复用，同时为程序的再工程和维护进行设计和编码遍历(navigation)。

可复用性特性可由模板文档化，一个文档化的模板为对象提供了下列信息：父类(parent classes)(一个继承层次上)；相关联的类；属性(attributes)目录；属性说明(属性的值域和值限定；属性是否可改变、关键、计算等)；对象(object)限定(布尔条件)；方法(methods)目录；规则(rules)目录。

5 相关的工作

程序分片技术尽管非常有用，但它仅提供程序之后的设计恢复概念的间接方法。为了能更直接地解决这个问题，必须对编程和应用域概念建模并将它们联接到相应的抽象设计和代码上，国外许多的研究项目将其作为理解程序环境所必需的部件。文[4]中的LaSSIE系统是用来解释一个复杂的应用设计以及确认潜在的可复用代码。在LaSSIE系统使用了一个基于框架的专家系统来存放领域模型、抽象设计和代码。在文[5]中，Rich等对自动程序的识别进行了探讨，并在一些系统中尽力定义用以连接抽象概念及其实现的程序规划库。在自动程序的识别过程中，规划可以以继承的方式组成，程序用来查找规划实例，识别过程由程

(下转第127页)

```
    ]
    j←⌊pool-top/m⌋+1
    i←pool-top mode m+1
    Pool2(i)(j)←New Task
    If top(i)<n then
        Top(i)←top(i)+1
    modified←true
End
```

事实上循环队列和一维池也都可以实现多台打印机打印的调度，但由于打印机为一台以上，所以此时要用一个队列来对打印机的任务申请进行管理，也就是要对打印机进行排队管理；另外也不能使打印机任务达到均衡，这会使得一些打印机不停地工作，而一些打印机则经常闭置，显然这对打印机资源有效利用不利。

4 基于循环队列和基于池的打印任务调度的比较

队列简单易行，但不支持优先级；新的打印任务只能加入到队尾。如果想调整打印任务的顺序，要手动进行修改。池比队列要复杂一些，但它的最大特性是支持优先级，新的打印任务自动加入到适当的位置。

由于循环队列没有循环语句，所以队列的时间复杂度为0；而池的结构的维护由其秩序维持程序进行，其中主要是快速排序，而快速排序的时间复杂度为O(nlog2n)。

两种结构的空间复杂度是一样的。

参考文献

1 Ford W，Topp W 著. 数据结构 C++语言描述. 第3版. 刘卫东、沈官林译. 清华大学出版社，1998. 11

2 Stallings W. Operation Systems Internals and Design Principles. 3rd ed. 清华大学出版社，1998. 6

3 Shaffer D A 著. 数据结构与算法分析. 张铭，刘晓丹译. 电子工业出版社

（上接第123页）

序描述的低层次抽象到高层次抽象进行，其许多的描述恢复可复用性技术是半自动的。如果我们把自动程序的识别的研究结果按真实程序的比例放大，那么将来的工具就有可能控制一定规模的再工程过程。

面向对象的程序更新研究与可复用的再工程是相关的。在文[3]中介绍了一种在C语言描述的程序中识别对象的方法，候选类的选择是基于类型定义的分析；其次，那些带有一个给定类型参数或一个给定类型的值的过程、可被看成是候选的方法。在文[6]中，则描述了过程化的程序通过渐增的再工程方式建立一个面向对象的体系结构。

结束语 许多研究者和实践者都认为软件的可复用技术将提高软件的研究效率，并从根本上改变软件的升级，从而促进软件产业的变革。通过软件复用，在应用系统开发中可以充分地利用已有的开发成果，消除许多重复劳动，从而提高软件生产率和软件质量。软件复用中的一些问题是与现有系统密切相关的，而软件的再工程正是解决这些问题的主要技术手段。

可复用性的再工程技术有三方面的益处：第一，原有系统的优点将被保存下来；第二，设计和代码通过系统给定的应用域能达到一致，易于将来的维护；第三，可复用框架使得再工程和其它新系统的开发更容易。

未来的工作将更集中于评估这种技术在实用的再工程项目中的运用，以及高层次的再工程各阶段的自动化处理。这些技术问题的解决，将涉及到用CASE工具联接领域建模和设计抽象，以及程序中概念识别的自动化。本文中描述的这个具有可复用特性的代表性框架，也可引入形式化规格说明方式加以改进，当然这需要基于其它许多先进的研究领域，包括领域建模、可复用技术及软件规格说明中的代码自动生成技术等的进展。

参考文献

1 Jarzabek S. Software reengineering for reusability. IEEE Software，1993(3)：100～106

2 Arnold S. Software Reengineering. IEEE Computer Society Press，1993

3 *Chikofsky E，et al.* Reverse Engineering and Design Recovery：A Taxonomy. IEEE Software，1990(1)：13～18

4 Devanbu P B，et al. LaSSIE：A Knowledge-Based Software Information System. CACM，1991，34(5)：34～49

5 Rich C，Wills L. Recognizing Program's Design：A Graph-Parsing Approach IEEE Software，1991(1)：82～89

6 Jackobson S，et al. Re-engineering of old system to an object-oriented architecture. In：Proc. OOPSLA'91，1991. 340～350

7 郭耀，等 再工程——概念及框架. 计算机科学，1999，5(26)：78～83

8 蔡希尧，陈平. 面向对象技术. 西安电子科技大学出版社，1993