软件重用的概念和方法

中国科学院计算中心 潘玉平 张 悦

一、引言

当前软件管理的主要注意力集中于如何降低软件开发和维护的代价。作为回答之一、可重用软件成为人们讨论的热门话题。

可重用软件的使用不仅能够提高软件的生产效率，而且能够通过改进可靠性、可移植性、可维护性、模块化程度、独立性、自说明性及可验证性等诸方面来提高软件的整体质量，同时，还能够减少开发时间和降低成本。

本文在介绍了软件重用的一般概念后，着重讨论用部件库方法实现个别部件(component)的重用。

二、软件重用的概念

1.什么是软件重用

软件重用的定义应当是足够广泛的，对于那些可以被收集来并用在以后开发其它软件过程中的任何信息的重用，都应当包括在内。

具体来说，软件重用包括对以下各种信息的重用：软件需求说明、系统设计、源代码模块、操作系统、文档、分析数据、测试信息、维护信息数据库以及软件开发计划和方法学。

对于下述项目的重用——用于生成软件的自动工具，用于分类和检索各种无错软件部件的定义良好的可重用软件体，以及用于改善软件生命周期各环节的集成化的软件支撑环境，都可以看成是软件重用工作的一部分。

因此，那种认为软件重用就是对现存源代码的重用的见解，是一种误解。

2.可重用软件信息的级别

软件重用实际上就是在新的场合下使用以前得到的概念或对象。它贯穿于需求说明，设计，编程及测试这一软件开发的整个生命周期。

可重用的软件信息可以划分为三个级别，表 1 对此进行了归纳。

表 1 可重用软件信息的级别

级别	例子
1.思想的重用 (Reuse of idea)	说明、设计、开发方法学和技术等
2.领域知识的重用 (Reuse of Domain Knowledge)	文档、技术手册、计划参加人员、分析数据、测试和维护信息数据库等
3.个别部件的重用 (Reuse of Particular Components)	源代码、子例序、模块、操作系统、软件包、编程语言和工具等

对表 1 所归纳的内容，现作如下补充说明。

(1)思想的重用。在软件工程中，软件需求说明，设计开发方法学是软件开发思想，它们可以被重用于建设新系统。

如果把系统说明到可执行实现这一变换过程记录下来，并且能够将它重现出来的话，那么需求和说明改变时，可以重用以前的开发，而只作微小的改变来生成新的实现。

(2)领域知识的重用。重用领域知识的最为普通的方式是对软件开发人员的重用，在每一次开发或修改软件应用时，存在于软件程序员头脑中的领域知识都得到频繁的重用。

(3)个别部件的重用。部件库和现存软件的重用是个别部件重用最普通的例子。关于部件库方法，将在下一部分详细进行讨论。

可重用软件信息的级别不同，重用的难度也就不同，源代码的重用也许是最困难的重用形式，而高级软件信息(即需求，说明和高级设计)的重用相对而言要容易一

些，因为前者比后者与硬件和操作系统联系得更密切。

3.软件重用的发展

迄今为止，软件已被成功地重用的场合有：小的软件项目，软件工厂方法。

(1)小的软件项目。小的软件项目之所以能够重用是因为：

a.由重用软件的那一个人写的。

b.同一项目中另一个人写的。

c.开发一个应用之前，已有了过去的版本或类似的程序可用。

d.软件功能具有如下功能：

- 已被良好理解
- 只有极少的数据类型
- 基于成熟的技术
- 在问题领域中已有标准

(2)软件工厂方法。日本已经采取一种与众不同的方法来编制程序。他们把编程看作软件生产，而不是软件开发。他们将编程生产效益提高的原因归纳为：

a.他们已经建立了大量的软件部件并培养了大批使用和开发部件的程序员。

b.他们把软件开发过程划分为各个阶段，并将不同阶段分配给软件工厂内的不同机构。

c.他们已经开发了一整套软件工具和标准来支持软件开发生命周期中的重用。

d.软件重用是他们培训过程的一部分。有的软件工厂每个月对所有的软件程序员进行一次编程练习。这些练习要求引用软件部件库以便用最小的代价来完成。

三、用部件库实现个别部件重用

1.个别部件重用的思想和实现过程

现举一个例子来研究个别部件重用的思想和实现过程。

本实例用经过选择的反应其功能特点的标准属性来描述软件部件。建立一种分类体制，该分类体制基于这样一种假设，即可用的软件通常并不完全满足需求，需要为其制定修改的规则，而不认为是意外情况。提供一种定位方法并评价它们各自的适应程度，重用过程如下。

a.给出一套所需部件的功能说明。

b.搜索部件库找出符合说明的候选部件。

c.如果某个部件符合全部说明，则它就是所求。

d.更为一般的情况是，存在多个候选者，每一个只满足部分说明，则把它们称为相似部件。在这种情况下，根据这些相似部件对需求的匹配程度进行排序，找出最佳候选部件进行修改，以使其符合全部说明。

e.将合适部件有机地合成在一起。

下面给出了一个较为形式化的适用算法。

```
Begin
  Search Library
  If Identical match
    Then Terminate
    Else
      Begin
        Collect similar components
          For each component
            Compare degree of match
      Rank Select best
      Adapt Modity component
        End
  Concatenate
End
```

2.个别部件重用存在的困难及可能的解决方法

一个软件部件能够被重用，在实现上还存在以下四方面的困难：

- 标识，定位和检索方面的困难
- 理解方面的困难
- 修改方面的困难
- 合成方面的困难

下面，将讨论上述四方面的困难及解决方法。

(1)标识、定位和检索。可重用软件之所以进展缓慢，是因为写一个程序比找一个程序容易得多，因此要解决这一问题，就要有一个好的说明方法来标识软件部件，即需要有一个好的分类方法，使得软件开发者在需要时能够在软件部件库中找到侯选软件部件，并与对部件的要求进行完全或最近匹配。

分类是将相像事物合成一组的艺术。组中成员至少具有一个共同特性。而其他组内成员不具备这一特性。分类应当能够表达组内成员之间和组与组之间的联系与

区别。

可以作为可重用软件的分类信息有:系统信息,功能信息和属性信息。

a.系统信息:表明该部件运行时所需的系统限制,如主机名称、操作系统名称及版本,屏幕管理软件名称及版本等。

b.功能信息:表明该部件运行时所执行的功能,如查找、显示、检索、管理等。

c.属性信息:表明部件本身的属性及连接约束,如创建者、创建日期、规模、部件号、部件名称、父部件号、子部件号等。

目前分类较为成功的有 Ruben Prieto-Diaz 为部件库开发的分类模式,Haberman 的用于 Gandalf 环境的 Program Schemata,数学软件库的 Gams 分类方法。

(2)理解。理解也是软件部件难于重用的一个重要问题,这是因为:

- 程序难读
- 程序的理解依赖于上下文
- 有关程序的信息难于寻找

不同程序员有书写程序的风格,好处是,程序员能充分发挥主观能动性,但也存在程序员之间彼此不易理解的问题。我们常常会有这样的感觉:读别人程序所花费时间比自己写同样程序要长。

注意:无论部件修改与否,程序必须被理解。

解决理解问题,重要的途径就是完善文挡。

作为一个工具,Hypertext 系统给解决理解问题指出了光明的前景。Hypertext 是一种管理信息的技术,也是一种电子化文档的形式,它将数据或信息存储在许多节点(Nodes)中,而节点用链(Links)连接成网状结构。在节点中的的数据可以是正文、源程序、图形、图象、动画、声音或它们的组合。用户可以用此工具建立一个完善的文档。

另外,Matsumoto 博士也提出了解决理解问题的一种方法,即对一部件进行多层次描述,并建立适当的跟踪功能。这种方法在日本的软件工厂中广泛使用。

(3)修改适用。当一个软件部件被重用时,它不可能完全满足目标的要求,为使其更好适合系统,就必须对部件进行适当的改进。

两种典型的修改是:

a.通过修改旧的实体(类型)来产生新的实体,比如通过增加功能把一个二分查找例程改为二分排序例程。

b.产生类型的新实例,举例来说,使用参数来给包含 Ada 类的软件实例化或产生特定的 Ada 类,这些参数使这些 Ada 类成为特别。通过改变参数来产生新实体比改变源代码更可取。

对旧实体的修改问题可以通过基于框架的软件工程及完备的修改工具来解决。

(4)合成。用已存在的软件部件构造新的系统,必然存在合成问题。

合成的两大原则是:信息转递(Message-passing)与继承(Inheritance)。部件合成前,要进行合成检查,被合成的部件必须满足一定的条件,以保证合成的一致性和完整性。利用模板在一定程度上可以合成问题。

模板是表达应用系统结构的模型,应用系统的设计也就是模板的设计,模板是一个不完全的树结构,它有一个根结点,但一般结点可有多个父结点,每个结点代表一个部件,部件之间是父子关系,即父结点可调用子结点,一个子结点可被多个父结点调用。

模板可以动态生成,也可以根据应用提前生成存于模板库中,动态生成模板可在生成工作台上通过人机交互生成,存于模板库中的模板可根据应用从模板库提出,裁剪,对模板裁剪有一定的限制,例如:

a.有些结点是必需选择的,如根结点。

b.如果父结点被删除了,则其后代所有结点就被删除了,除非它(它们)还有来自其它路径的父结点。

c.有些结点有不止一个父结点,那么,仅当其所有父结点都被删除时,才可将其删除。尽管软件重用的前景非常乐观,但只有不断地实践才能使其实用化,此外软件重用除了上述四方面实现的困难外,还有许多第二位的问题,像效益问题,平衡问题,甚至可重用软件的数据权利和义务的法律问题,这些问题也不能忽视。

参考文献:

1."Classifying Software for Reusability", IEEE Software, Jan, 1987

2."Managment Guide to Software Reuse", NBS Speacial Publication 500-155

▲