使用可重用构件合成软件

李文军　李师贤　（中山大学　计算机科学系）

摘　要

This paper serveys some existing software construction methods from the viewpoint of reusability and describes four techniques for software construction from reusable components, function-based technique, data-based technique, object-oriented technique and application-oriented technique. Their advantages and disadvantages are also discussed.

软件重用被认为是软件工程中极有前途的发展方向，对于提高软件生产率和软件质量有重要意义。目前，软件重用研究工作主要从方法学、工程和技术三方面进行。

软件重用方法学主要为软件重用工程提供理论上的指导，它包括研究软件重用生命周期模型、软件重用内容、软件重用对象以及软件重用的经济因素，心理因素和社会因素等。

软件重用工程从软件开发的整个生命周期来研究软件重用问题。它基于某个软件重用生命周期模型，该模型以可重用软件库为核心。软件重用工程的主要问题是软件的标准化、软件特征的提取和可重用软件库的组织等。

虽然软件重用方法学十分强调软件重用不仅仅限于源代码一级，但是现有的大多数软件重用技术都是面向源代码一级的。这些技术主要有可重用软件的构造技术、合成技术、查找技术，理解技术和修改技术等。

一.可重用构件合成技术

我们称存放在可重用软件库中的软件为可重用构件（reusable component），简称构件。构件可以是规格说明、设计、源代码、测试数据或文档等。本文主要讨论源代码一级的可重用构件合成技术。讨论基于图1所示构件重用模型。在该模型中，构件合成技术研究的是如何根据新软件需要提出对构件的需求以及如何将获得的构件合成到新软件中。模型中需要用到的其他技术不在本文中详细讨论，当构件合成技术和它们发生联系时再加以说明。

```
BEGIN
  给出构件需求R;
  根据R查找构件库;
  IF  找到符合需求的构件  THEN
    BEGIN
        令C为找到的构件
    END
  ELSE
    BEGIN
      找到相似构件集S;
      FOR  S中每一构件C(I)
      DO  计算C(I)的匹配度;
      选择最佳匹配构件C';
      修改C'为C使之适应构件需求;
      IF  C可重用  THEN  将C加入构件库
    END
  ENDIF
  将C合成到新软件中;
END;
```

图1　构件重用模型

根据构件合成技术所采用的构件组织方式可以将它们分为四类：基于功能的构件合成技术、基于数据的构件合成技术、面向对象的构件合成技术和面向应用的构件合成技术。这些技术都有自己的适用领域。

二.基于功能的构件合成技术

对于以子程序技术（过程、函数、例程

等）构造的可重用构件，比较多地采用基于功能的构件合成技术。构件合成的结合方式主要是通过子程序调用和参数传递。这样的合成方式在参加合成的构件数目较少、构件接口说明清晰的情况下非常有效，例如数值计算软件包、图形软件包等应用是很成功的。

采用这种技术设计新软件时，首先基于功能对软件系统进行分解，系统被分解为许多相互间耦合度低而内部聚合度高的模块。对可重用构件的需求根据模块所实现的功能提出。

UNIX操作系统提供的管道（pipeline）机制和输入输出重定向（I/O redirection）机制也可以实现构件合成[1]。如果需要现有的两个程序临时合作完成一项任务，但又不值得重新编制一个新程序时，可以通过管道机制重用这两个程序。例如UNIX的shell已有两个实用程序ls和wc，其中ls列出目录文件，wc统计文件行数。现在要在列出目录后知道文件的个数，这时不必构造新的程序，使用管道可以把ls和wc合并在一起工作：

```
ls／usa／kc  |  wc  -l
```

其中，“|”是管道连接的特殊shell记号。

使用输入输出重定向也可以完成以上任务，不过会产生中间文件，程序的连接语法不如使用管道清晰。管道和输入输出重定向都是基于UNIX操作系统的标准输入和标准输出，可以看作是子程序调用和参数传递的扩充。

对于参加合成的构件数目较多、构件组织方式复杂的情况采用基于功能的合成技术就难以保证合成结果的结构的正确性。基于数据的合成技术可以克服这个缺点。

三.基于数据的构件合成技术

基于数据的构件合成技术首先根据所处理问题的数据结构构造出一个框架（Framework），然后根据问题需要列出所需构件清单，逐一把构件注在框架中的合适位置。

Jackson方法是基于数据的设计方法的一个代表，JSP（Jackson Structured Programming）可用来开发规模不太大的软件系统[2]。我们在微机上实现的程序设计环境ZJPE支持以JSP方法进行程序设计[3]。ZJPE扩充了一个可重用构件库及相应管理机制，使环境还可以支持构件重用。ZJPE构件重用过程如图2所示。

ZJPE的可重用构件库中存放的构件有两种：一种是程序设计语言提供的基本操作；另一种是用JSP方法或其它软件开发方法编写的模块，称为“宏”。ZJPE帮助在库中查找构件和增删库中构件。

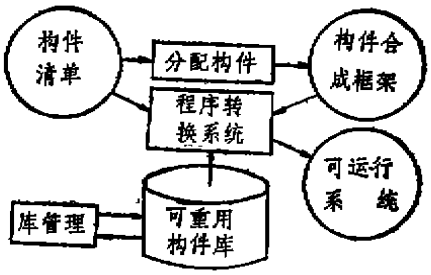

图2 ZJPE构件重用结构

ZJPE用户（即软件开发人员）使用环境提供的结构图编辑器和程序结构生成器等工具构造出程序结构，程序结构被看作是合成构件的框架，以JSP结构图表示。用户列出所需构件清单（JSP方法中称为操作表），然后将构件逐一在图形方式下分配到框架中。分配时用户只操作构件清单上的编号，ZJPE在库中取出相应构件合并到框架中的指定位置，然后转换为结构正文，自动生成可运行系统。

基于数据的构件合成技术的最大特点是所合成的结构与问题结构相对应，开发人员把精力放在结构的设计而不是控制流的设计上。这样生成的软件结构清晰，易于测试和维护。

ZJPE中所不足的是将构件分配到框架中时，构件结合的方式仍然是传统的子程序调用和参数传递。在上述构件重用模型中，如果只能查到相似的构件时，这种结合方式

与面向对象合成技术的继承机制相比较就显得不够灵活。此外，ZJPE的可重用库中的构件构造过程没有融入到环境中，因而构件的修改得依赖于其他工具。

四.面向对象的构件合成技术

不同的面向对象程序设计语言有不同的概念和术语，这里我们选择Eiffel中的概念和术语作为描述语言。Eiffel是目前商品化的功能比较齐全的面向对象程序设计环境[4,5]。Eiffel是程序设计环境的名字，也是程序设计语言的名字。

Eiffel对系统的分解既不是基于功能也不是基于数据，而是基于对象。对象的外部属性用抽象数据类型描述为类（class），即使系统的功能需求发生变化，对象也很少改变。一个用Eiffel开发的系统是对象的集合。

在Eiffel中可重用构件是描述对象的类，程序设计环境提供了一个可重用构件库——标准类库。Eiffel使用两种机制支持构件合成：顾客（client）和继承（inheritance）。

如果在类A中包含形如bb：B的说明，则称A为B的顾客。A通过B的规范中定义的特征使用bb。特征包括属性（数据）和例程（操作）。

继承是面向对象语言的特色之一，也是强有力的构件重用机制。当一个新的类被说明为一个已定义类的后继时，新类就包含了父类的所有特征和相关性质。继承使得软件开发成为一个进化的过程。

如果类A是类B的一个顾客，则A只使用B的说明；如果A是B的直接后继，则A可以直接依赖于B的实现，这时信息隐藏原则不起作用。这两种重用方式是互补的。

面向对象方法的特点是使可重用构件的构造技术、修改技术和合成技术揉合在一起，这使得软件重用更加方便。Eiffel有两种方法修改所继承的特征：通过重新命名（rename）进行语法上的修改，或通过重新定义（redefinition）进行语义上的修改。这样的修改技术保证了重用构件既能满足新需求，又不会影响到原有构件。

与基于功能或基于数据的合成技术相比，当找到的构件需要进行修改时，面向对象合成技术具有优势。目前的大多数构件重用是需要修改原有构件后才能满足新需求的。

在ZJPE环境和Eiffel环境中都没有强调构件的分类与检索。当构件库比较小的时候，这不会产生什么影响。但是随着库中构件数目不断增加，用户检索构件变得越来越困难。这是软件重用工程所要研究的问题。

五.面向应用的构件合成技术

Draco系统是使用面向应用的构件合成技术的一个例子[6]。在Draco中软件构造始于对应用领域的研究，分为三个阶段：（1）确定分析领域；（2）进行领域分析；（3）进行领域设计，产生领域描述。对于Draco中已有的领域中的系统开发，就可以重用领域分析员和领域设计员的成果——领域描述。

Draco的特点是允许可重用构件是"分析"或"设计"。Draco中定义了一类领域后，系统分析员通过已有的领域考虑用户的新需求，如果存在一个领域可以满意地描述新系统的对象和操作，则系统分析员就有了一个框架，可以将新规范挂在框架上。这样就实现了分析的重用。系统设计员通过与Draco交互将问题求精为可执行代码，交互过程中可在Draco领域设计员给出的不同实现方案间作出选择。这是设计的重用。

在其他技术中要做到分析和设计的重用是困难的，因为问题领域的不确定使得领域分析难以进行，分析结果难以组织，设计更难考虑到众多的实现方案。

然而，面向应用构件合成技术局限于某一特定领域的重用。它要求这一领域的结构已完全被人们所了解，并要求具有该领域丰富经验的领域分析员和领域设计员进行领域

从Smalltalk语言的结构特征看“软插件”的形成

朱海滨　陈火旺　（国防科技大学计算机系）

软件的可重用性是软件工程领域中的一个重要课题，现已经有许多人在探索软件可重用的途径，面向对象的程序设计方法则是以解决软件可重用性问题作为自己的重要目标之一。另外，Smalltalk语言经过十几年的实践，走向了成熟，开始为广大程序人员所欣然接受，国内已有相当数量的软件工作者开始研究该语言。我们在VAX-11/780机上用C语言实现了Smalltalk-80系统的核心环境，深受Smalltalk语言其独特构造的启发。本文从Smalltalk语言的结构分析出发，提出了“软插件”的概念，并探讨了此概念形成和实用的可能性，以及它在软件可重用领域中的实践意义。最后，讨论了面向对象方法与“软插件”概念的必然联系。

一、 Smalltalk语言的结构

作为运行环境，Smalltalk语言由两部分组成：即虚拟机(Virtual Machine)和虚拟象(Virtual Image)。虚拟机是一个虚设的用软件实现的面向Smalltalk语言的“硬设备”，用于支持虚拟象的存在并解释虚拟象的功能实现。若单从语言结构上看，Smalltalk语言的结构特点完全反映在其虚拟象上。因此，下面只对Smalltalk语言的虚拟象进行讨论。

1. Smalltalk语言的构成元素——类

如果将Smalltalk语言比作一幢楼房，那么类就是建造这座楼房的砖石。Smalltalk语言的虚拟象大约由五十多个类组成，该语言的各种功能分别由这些类给予支持。类作为

分析和设计。这在很大程度上限制了重用的潜力。

软件重用的许多工作都有待我们去研究和解决，可重用构件的合成技术只是其中的一个方面。从软件重用工程的角度看，构件合成技术的成功与否还取决于与其他技术的集成化使用，如构件构造技术、修改技术、理解技术等。一个支持软件重用的系统应是以可重用构件库为核心的集成化支撑环境。

尽管软件重用日益显示出其巨大潜力，但离实际的大规模重用还有很大的距离。国内外计算机科学家对软件重用的研究动向和研究成果值得我们借鉴。

参考文献

〔1〕 K.克里斯琴(孙玉方，董士海译)，'UNIX操作系统'，John Wiley & Sons Inc.，1983

〔2〕 M. Jackson, 'Principles of Programming', Academic Press, 1975

〔3〕 李师贤，李文军，阮文江，吴红，'程序设计环境ZJPE'（待发表）

〔4〕 B. Meyer, 'Eiffel, A Language and Environment for Software Engineering', J. Sys. Softw. 1988, 8

〔5〕 B. Meyer, 'Reusability, The Case for Object-Oriented Design', IEEE Softw. 1987, 3

〔6〕 J.M. Neighbors, 'The Draco Approach to Constructing Software from Reusable Components', IEEE Trans. Softw. Eng.1984, 9