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

一种语言"构件"，具有以下外部特征和内部特征。

从用户的角度来看，一个类的外部特征是：1）类是一组具有相同或相似性质对象的统一描述；2）类由三个部分组成：对象的数据结构描述、对象可接受的操作描述、类的继承性描述；3）类是生成对象的统一模板。

从实现者角度来看，一个类的内部特点是：1）类是一种代码共享手段，由它生成的对象均共享其功能代码的实现；2）类由以下项目组成；Ⅰ.标识及注释：用于标识该类，指出该类的名称，说明该类的功能和结构特点；Ⅱ.继承说明：描述该类对象继承的类，被继承类的性质均为该类对象所持有；Ⅲ.数据结构描述：描述该类对象所应持有的存贮空间的组成和结构；Ⅳ.操作实现：是对象所能接受操作的具体代码实现，它包括类的外部特征中描述的操作和该类对象的许多私有操作；Ⅴ.与其它类的关系描述：用于系统组织和内部控制。

2. Smalltalk语言中类的组织

类的设置使得Smalltalk语言的实现非常简单，易行。Smalltalk语言其实就是其所有类功能的有序集中。这些类按以下原则组织：

1）语言中所有的类在概念上按其继承关系形成树形结构（图1）。

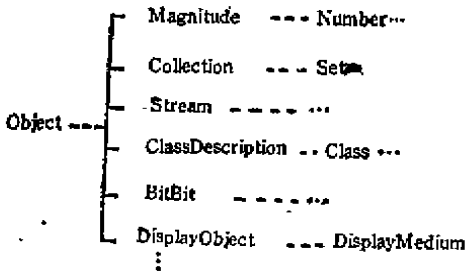

图1 （图中用线相连的上层类称为下层类的超类）

2）每个类可作为一种特殊的对象来看待，这类对象的共同特征由Class类来描述。

3）每个类都是Object类的子类（直接的或间接的）。

4）每个元类对应于一个类，用于支持这个类作为对象应具有的行为实现。

5）每个类都是其元类的实例。

6）每个元类都是Class类的子类（直接的或间接的）。

7）每个元类都是MetaClass的实例。

8）Class类和其超类支持所有类的共同行为。

（以SmallInteger类为例，语言的组织如图2所示）

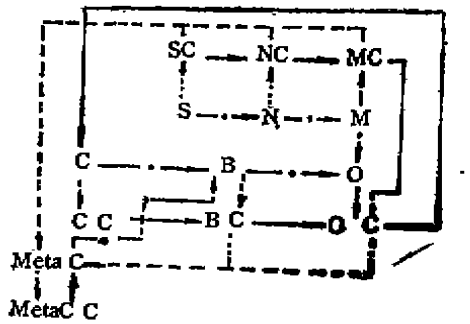

图2（S=SmallInteger，C=Class，M=Magnitude，N=Number，B=Behavior，O=Object）

其中，"—·→"表示类的直接继承关系，A—·→B表示A直接继承B；"→"表示元类的直接继承关系，A→B表示元类A直接继承元类B；"—→"表示实例关系，A—→B表示A是B的实例。

3. Smalltalk语言的构造特点

综上所述，可以发现Smalltalk语言在构造上具有以下几个特点：

1）Smalltalk语言由许多相对独立的单元——类组成；

2）Smalltalk语言的功能可以根据需要通过增删类进行扩充或缩减；

3）Smalltalk语言的组织和连接简单，主要靠继承关系和一组外协消息联系；

4）Smalltalk语言安全，可靠，易于检错，因为类的故障不会传播和扩散，易于维护。

二、类与插件的比较

计算机的硬设备是由许多插件板连接而成的，而这些插件板又是将许多具有独立功能的集成电路插件（图3）按插件板的设计要求组装连接而成的。要构造计算机系统的

硬设备，制造人员不必透彻地了解每个集成电路插件的内部结构及实现，只要了解每个插件的使用要求，功能说明和对外接头的作用等外部特征，便可以按要求在设计好的插件板上安装插件，进而构成整个系统的硬设备，这样，构造计算机硬设备的工作主要是插件板的设计。计算机的迅速发展与这种构造特征及集成电路的发展是分不开的。

在软件工程中要提高软件生产率，就应当走类似硬件的道路，应该有软件的“集成电路”。Smalltalk语言的构造方法以及类的实现可望为软件系统的构造开辟新的途径。从类的外部特征来看，它可以作为一种“软插件”（图4）。它具有一组外接插头：继承描述和一组外协消息，以及随之提供的说明书。依Smalltalk语言的构造策略，要构造一个软件系统，如果系统所需要的类均已存在，构造者只需要对整个系统的构造策略和原则进行精心描述和制定（这相当于硬设备插件板的设计），将所需的类按这些原则组织起来（这相当于将插件装入插件板），便可以推出一个新的软件系统，这样，软件生产率将会获得很大的提高。

图3

图4

三、“软插件”的提出

研究软件可重用性的目的就在于尽可能地在软件生产中使用现有的软件，而尽量少重复编写具有类似功能的软件。据统计，在生产一个新的软件系统过程中，大约80—85%的工作量耗费在重复编写已经编写过的软件单元上，因此，必须寻找一种能够比较容易地将已经成熟的软件单元应用于新的软件系统中的技术。最好有一种集成机制，可以将已经成熟的软件单元制成一个个相对独立的实体，使它们可以不加改动或很少改动就可以应用于新的软件系统中，只有这样，软件生产才有望减少重复劳动，提高生产率。这里提出的“软插件”概念就是基于以上思想的一种软件集成机制。

目前，软件可重用性的研究一般分为四个层次：代码的重用，设计的重用，工具的重用和环境的重用。“软插件”可以依据不同的“集成度”以适应不同层次的重用，很明显，“软插件”的集成度越高，所支持的可重用层次就越高。

“软插件”作为软件的一种集成机制，必须具有以下特征：

1）模块性好，独立性强。一个“软插件”应是可以独立存在的实体，它应当不受或少受外界的影响，以便能够较为自由地为各个不同的软件系统所使用。

2）可靠性好。因为“软插件”是用于构造新的软件系统的基本单位，因此必须保证它具有很好的可靠性。

3）连接简单，使用方便。为了方便并安全地建造新的软件系统，必须要求“软插件”之间的连接简单，特别是由于“软插件”目前的连接尚未有成熟的“硬化”技术，因此更加要求其连接简单，才能达到减少工作量的目的。

4）封装功能。作为集成机制，“软插件”必须将被集成的功能实现封装起来，以便使用户不必搞清楚它的内部细节，从而加快软件工程的进度。

5）内部功能的高效实现。“软插件”是构造软件系统的基本单元，其内部代码实现的高效与否直接影响到软件系统的性能，这一点应当是对“软插件”的基本要求。

6）清晰，简洁的说明。“软插件”是作为产品交给用户的，它应当具有类似于硬件集成插件的功能及各项指标的说明，以便用户可以根据需要选择合适的“软插件”。

通过以上分析，我们可以看出，“软插件”为软件可重用性的研究提供了一种新的途径，“软插件”的实践将会为软件生产率的提高和软件设计方法学的研究起到较大的推动作用。

四、 类作为“软插件”的雏型

从Smalltalk语言的构造中，我们可以看出，类已经开始充当了系统构造的单元，

并在实践中取得了初步的成功。之所以这样，是因为它具有以下几个特征：

1）类是一类对象的统一模板，它具有很强的模块性，类的功能代码实现只通过对外接头与外界联系，具有很强的独立性。

2）类是对象的一级抽象，它将一类对象的数据结构描述和功能实现封装起来，使得外界不必清楚其内部实现，只要从其说明中了解该类的基本功能就可以使用它。

3）类之间的连接只有继承性描述和一组外协消息，清晰，明了。

4）类的可靠性表现在一个类的出错不会传播到其他类中，一个系统的类如果出错，可以将其删除，作为一个降级的系统使用，不会使整个系统瘫痪。

另外，类支持代码的共享，类中的代码均是可再入的，这两点也支持了类的可重用性。因此，类从概念上到实践中都证明了它是有资格作为"软插件"来组织软件系统的。

五、 对类作为"软插件"的评价

Smalltalk语言运用类在构造软件系统的方法上做了成功的尝试，但是，类要完全合格地担当"软插件"的角色还存在以下一些不足之处，仍需进一步改进。

1）在Smalltalk语言中，类还没有被作为唯一的语言构成单元，还有元类与类的区别，影响了"软插件"概念的统一。

2）类的继承描述是一固定的插头，由它确定了类之间的线性依赖关系，影响了系统组织的灵活性。

3）Smalltalk语言中的继承只有一种，不能完成多重继承，影响了系统的代码共享。

4）在Smalltalk语言中，类作为"插件"只是形成了软件概念，还没有形成真正方便实用的"插件"。

六、将类改进成为更实用的"软插件"

经过以上的讨论，我们可以设想从以下几个方面入手对类进行改进：

1）取消元类与类的区别，在系统的构造中只要求类作为基本单元，从Smalltalk语言的实现中，我们认为，这是完全可以做到的。

2）设置类的多重继承机制，增强其代码共享能力，这样做不会影响使用的难度，类的对外联系仍很清晰。

3）建立类的标准，从语法上统一类的实现规格，以增加类的兼容性。

4）进行类插件语义等的形式化研究，从理论上确立"软插件"的可行性。

5）研究"软插件"的物理支持机制，建立"软插件"的存贮和传送机制，使类真正成为实用的插件，可以灵活快速地应用到新的软件系统中。

七、 "软插件"与面向对象的设计方法

Smalltalk语言是一种面向对象的程序设计语言，它以类作为基本构成单元，又以类作为程序设计的基础，因此，"软插件"是直接支持面向对象的方法的。

我们先看一下Smalltalk的程序是怎样组成的。

程序 ::=程序段 | 类创建.程序段 | 程序.
　　类创建.程序段

程序段::=对象创建.消息序列 | 程序段.对象
　　创建.消息序列

对象创建::=类 创建消息.

消息序列::=对象 消息 | 消息序列.
　　对象 消息

（其中，消息是程序员所见类的对外接头）。

根据以上的程序定义，用Smalltalk语言进行程序设计过程如图5所示。Smalltalk语言的程序设计风格是面向对象设计方法的具体实现，类的引入又使该设计方法中的对象定义非常简单，"软插件"的应用将成为面向对象方法得到推广使用的重要条件。

面向对象的设计方法可分为三个步骤：第一是对问题的定义和描述；第二是提出系统形成的非形式策略；第三是策略的形式化，包括标识对象及其对象的属性，标识对象上的操作，建立对象界面并实现其操作。而面向对象设计方法的特点则主要反映在第三步上。

"软插件"的引入使面向对象的设计方法更为方便和实用。下面我们给出基于"软插件"的面向对象的设计方法，这一方法可以不断地充实现有的"软插件"库，使得该方法的实用性不断增强。这一方法可以描述如

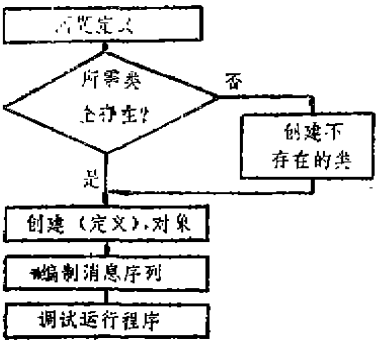

图5（*这一步的工作要比一般的程序设计简单得多，它的工作主要是建立对象间的联系）

下：

1）分析问题，进行问题的定义和描述；

2）制定软件系统的构造策略，也就是进行系统对象的划分和定义，描述对象间的联系；

3）进行对象的标识及实现，应当尽可能地使用库中现有的"软插件"，若库中没有相应功能的"软插件"，则要建立之并装入"软插件"库中；

4）使用第三步获得的"软插件"创建对象，并按第二步定义的策略组织系统；

5）建立系统对象间必需的连接，以协调整个系统的工作；

6）运行调试，提交产品。

Smalltalk语言的结构具有独特的风格，利用Smalltalk语言的构造方式，可以提高软件系统的构造速度，增加软件系统的可靠性和灵活性，这种风格直接支持面向对象的设计方法。同时，Smalltalk语言的基本构成单元——类已经具备了大部分"软插件"所应具有的特征，并取得了初步的成功。虽然这个概念还不够成熟，但我们相信，"软插件"的研究和实践必将推动软件可重用性的研究，并且在提高软件生产率方面起到重大作用。

主要参考资料

1. Adele Goldberg and David Robson, Smalltalk-80: The Language and Its Implementation, Addison-Wesley, Reading, Mass., 1983.
2. Stephen T.Pepe, Adele Goldberg and L. Peter Deutsch, "Object-Oriented Approaches to the Software Life cycle Using the Smalltalk-80 System as a Case Toolkit, "1987.
3. Bhaskar K.S., J. K. Pecol and J. L. Beug, "Virtual Instruments, Object-Oriented Program Synthesis." Proceedings of the ACM Conference on Object-Oriented Program Systems, Languages and Archetectures (OOPSLA), 1986.
4. Jim Anderson and Barry Fishman, "The Smalltalk Programing Language," BYTE, MAY, 1985.
5. Geoffery A. Pascoe, "The Elements of Object-Oriented Programming," BYTE, Feb., 1986.
6. Alan Snyder "Encapsulation and Inheritance in Object-Oriented Programming Languages, "OOPSLA'86 proceedings.
7. Stephen S.Yau, Jeffery J. P.Tsai, 软件设计技术之概观，计算机科学，4，1987，pp.10-18.
8. 杨芙清等，软件工程支撑环境中的集成化问题研究，计算机科学，4，1987，pp.1-10.

（上接第77页）

主要参考文献

1. Sleeman & Brown(ed), "Intelligent Tutoring System" Academic Press 1982
2. Feigenbaum & Barr(ed), Application of AI Research:Education in"AI Handbook" V.2 pp 225-290 1982
3. Anderson J. B. & al., Geometry Tutor proc. of JJCAI-85 1985
4. Brown J. S., Burton R., Clancey W., Application of AI to education and training No.2 Tutorial of AAAI-84 1984
5. Dede J., Review and Synthesis of ICAI, Int. J. of man machine studies V. 24 pp. 329-353 1986
6. Begg M., Intelligent Authoring System Proc. of computer-aides technology pp. 617-8. 1985