面向对象基本概念与语言

麦中凡（北京航空航天大学）

摘要

面向对象技术已经成为计算机技术的不可分割部分。采用这个技术，在软、硬件开发和人工智能研究中都显现出巨大的优越性。但同时各种流派的面向对象设计的出现使人眼花缭乱。本文试图讨论面向对象程序设计的基本概念并介绍面向对象程序设计语言或系统的发展概况。

对象、类、消息的程序设计模式其基本点在于对象的封装和继承性。封装使对象外界面和对象实现分离。要继承就得将对象管理起来并建立类体系。以及由此带来晚聚束和实体的多态性构成面向对象基本特征。文中论述这些特征有利于当前软件技术所追求的模块性、数据隐藏、数据抽象，可修改、易维护，并适宜于与用原型开发等目标。

文中从模仿Smalltalk、采用面向对象思想、可作面向对象设计三个方面简述了20余种基于对象的语言。

1. 引 言

自从80年面向对象语言Smalltalk及其环境向计算机界推行以来，面向对象技术引起了计算机界极大的重视。由于面向对象技术对于软件工程学濒临的困境和人工智能所遇障碍都是一个很有希望的突破口，所以十年来面向对象技术的研究遍及计算机软硬件各个领域：面向对象语言、面向对象程序设计方法学、面向对象操作系统、面向对象数据库、面向对象软件开发环境、面向对象硬件支持一目前已经取得丰硕成果[1]。虽然有些软、硬件产品已投入实用，但研究依然在继续，特别是在面向对象的理论方面。由于面向对象技术基本上属于工程学方法论的范畴，故什么是面向对象？怎样才能称之为面向对象设计？怎样的语言才是面向对象语言？一都没有准确定义。正如川菜好吃，各种"正宗"、"真正"川菜馆纷纷出笼，似乎只要是"麻""辣"菜都是川菜，面向对象技术也正面临这种情况。正因为如此，面向对象的概念不易于掌握，特别是熟悉了过程型程序设计的计算机工作者，对新的又非一致的编程风格很不习惯，它不象"结构化程序设计"那样易于为人们理解，所以，不象想象的那样很快地普及。在国外，从它诞生（1972年Smalltalk初版）到引起重视（1980年），到进入实用和成为计算机工作者不可缺少的手段（1986—）已经历了14年。自85年以后，我国介绍面向对象技术的文章相继出现。近年由于Smalltalk语言、C++语言软件微机版本的引入，面向对象技术的研究进入到一个新的阶段。面向对象技术已成为广大计算机工作者感兴趣的论题。

本文拟就传统的软件工作者学习面向对象语言和编程风格，谈谈面向对象程序设计的几个主要概念，并简略介绍面向对象程序设计语言发展概况。

2. 对象与面向对象程序设计

客观世界的问题都是由客观世界的实体及其相互关系构成的，我们把客观世界

的实体称之为问题空间的对象。显然，“对象”因问题的特殊性是不固定的。一本书可以是个对象，一家图书馆也可以是一个对象。

本质上，我们用计算机解题是借助某种语言规定对计算机实体施加某种动作，以此动作的后果去映射解，我们把计算机实体称之为解空间的对象。显然，这个对象因语言而异。例如，汇编语言提供的对象是存贮单元（地址码）；过程语言是各种内定义类型的变量、数组、记录、文件；LISP语言是原子和表。一旦提供了某种解空间对象，就隐含着该对象允许的操作，例如，布尔变量只能进行布尔操作，存贮单元只能作拷贝（取）、修改（存）操作。而且，在一个程序中这种操作是共享的。例如，不同的布尔变量都可以作布尔加法。

对象及其操作从动态的观点看就是对象的行为。问题空间对象的行为是极其丰富多采的，而解空间对象的行为极其死板且无特殊性（共享操作）。因此，只有借助于极其复杂的算法才能操纵解空间对象而得到解。这就是常说的“语义断层”，也是“程序设计”始终是一门学问的原因。人们只能在程序语言提供的对象和操作的概念上作程序设计。愈是高级的语言提供的概念愈多，愈是远离机器和过程实现的概念则愈宜人。

面向对象语言提供的“对象”概念，是让程序员自己去定义解空间对象。他先描述出对象及其行为，然后以传统的办法实现它。例如，数据堆栈可作为一个对象，它必须有一个栈体存放数据，可按后进先出的次序压入、弹出数据。满了不能压入，空了不能弹出。至于栈体放入什么类型的数据不是堆栈行为的主要特征。学过Pascal语言的人都会实现堆栈：定义一个数组式链表作栈体，再写PUSH，POP，IS-EMPTY，IS-FULL四个过程。不同的栈体数据结构，过程体不相同，但其行为是一样的。不过Pascal不能提供独立的栈对象概念，因为它的栈体数据结构不仅允许这四个操作，还允许其他任何对该数据结构可行的操作，这样就保证不了后进先出的栈特性。例如，在该Pascal程序中多写了一个排序过程，Pascal就无法保证它不执行。

面向对象提供一种机制使得私有的数据有其私有的操作，即描述这个对象（数据）的独特行为。这样就向着减少语义断层的方向迈进了一步。在某些问题上可以直接模拟问题空间的对象。这样写出的程序易于理解和维护。

从存贮的角度看“对象”，它是一片私有存贮，其中有数据有操作（例程或单个指令）。其他对象的操作不能直接操纵该对象的私有数据，只有对象私有的操作可操纵它。

从对象实现机制看，“对象”是一台自动机，其中私有数据表示了对象的状态，该状态只能由私有的操作改变它。Smalltalk[2]把操作叫做方法（method），Ada类语言用过程或函数实现操作。

每当需要改变对象的状态时，只能由其他对象向该对象发送消息（message）（Ada[3]类是过程或函数调用）。对象响应消息后按照消息模式找出匹配的方法，执行该方法（Ada类是执行过程或函数）。要注意，发送消息和过程调用的意义是不同的。发送消息只是触发自动机。同样的输入参数可因自动机状态不同得出不同的终态（如果输出则有不同结果），而过程调用时只要相同的输入，输出总是恒定的。因而Ada类语言仅仅是“借用”过程调用机制来实现消息发送。此外，对象中的操作，除了改变状态而外，也有使状态可见（例如其他对象可令其打印）的操作。

至此，我们对“对象”的初步概念是：

（1）它是计算机实体，有一个名字代表被封装的数据与操作。

（2）数据描述了对象的状态。

（3）有私有操作，可操作私有数据改变状态，每当其他对象发出消息，本对象响应时操作得以实现。

如果一个程序系统只有对象和消息两个概念，程序设计就是建立一个彼此能发消息的对象集合，我们说这就是面向对象程序设计。否则，除对象和消息而外还有其他概念（如子程序、过程等），即使提供了对象描述机制（如Ada的程序包，Modula-2的模块），也只能说是基于对象的（Based-Object）程序设计。

3. 类与实例

在面向对象程序设计之中，封装的对象是程序的基本单位，相似的对象可以和传统语言中的变量与类型关系一样，归并到一类（Class）中去。程序员只需定义一个类对象就可以得到若干个实例（instance）对象了。类是实例对象的样板，是数据的抽象。它所定义的数据集比常规语言的类型所定义的数据集要复杂一些，例如，Pascal定义的类型仅限于在内定义的操作前提下定义数据的结构或尺寸。用户在定义枚举类型、记录类型时并没有为其定义额外的操作，而类定义则可定义操作集（如前所述数据堆栈的四个操作）。在生成实例时也不象传统语言那样作个声明 Variable-1: Some-type，就可以了，而是事先规定一个生成实例的操作（Smalltalk是new，C++[4]是Constructor构造算子）。与此对应还有当实例在程序中不再使用时有Deconstructor解构算子，或无用单元自动回收。这样，类就不象类型，它本身也是一个对象，可以接受消息生成实例，类型只是数据的抽象描述，本身不能参与运算。

实例承袭了类中的数据和方法（操作），只是各实例的数据初始化状态不同。

4. 类体系与继承性

每个对象都是某个类的实例，一个系统中类对象是各自封闭的。如果没有继承性机制，则类对象中数据和操作就可能出现大量重复，例如，前述的堆栈，若有整数，字符，实数三种堆栈，它们的行为是相似的。我们把共同部分先定义为抽象的堆栈STACK，它对抽象的栈元素ELEMENT作 IS-EMPTY，IS-FULL，PUSH，POP操作。然后定义INT-STACK，CHAR-STACK，REAL-STACK，它们是STACK的子类，反之STACK是它们的超类。子类继承直接超类的全部数据和操作（变量和方法），只要在子类中把ELEMENT重定义为整数、字符、实数，并为之各写出一个打印（PRINT）的方法。新增加的数据或方法若与超类中数据和方法重名，则覆盖掉超类中的定义，否则是新增的。这样，我们只要在原有类的基础上修改增补少量的数据或方法，即可得到所需要的子类。然后生成其大小和初态可以不同的实例。

这样，越是超类越抽象（有更多的共性），越是子类越具体。比如，每个类都要生成实例，其生成方法NEW我们并未在STACK中定义，它继承自系统提供的最抽象的类OBJECT。于是面向对象中所有的类对象都处于一个类体系之中（见图1）。每个类对象都可以生成若干实例对象（可以看作垂直于纸面长出并与某圆圈一样的圆圈簇）。

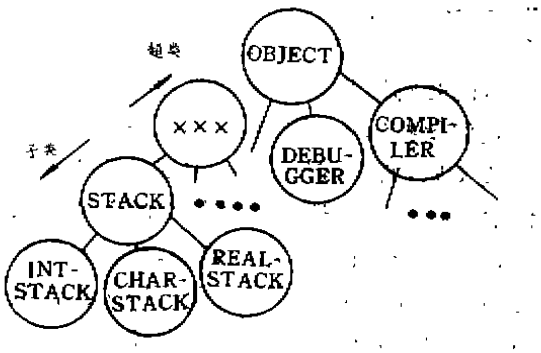

图1 单继承的类体系

Smalltalk中连编译器、非法程序等系统程序都作为一个对象统一地挂到Smalltalk的类体系中。

继承性符合软件可重用目标，只要运行过的程序，除非删除，都一直保留在系统中，越是后来，编的程序就越少。事实上，Smalltalk的图形窗口界面，用户只使用鼠标器就可以画出各种图形。

继承性体现了系统对对象的管理。有些语言虽提供了对象描述机制，但无继承的类

体系，其对象管理由用户建立自己的管理系统(通过文件)。继承性是面向对象系统的主要特征之一。

上述类体系是Smalltalk型的单继承体系。读者也许想到如果任何一结点可继承多个结点的数据和方法，拼拼揍揍后就是一个程序，岂不是更简单吗？即继承体系不是严格的树而是网。事实上，美国以LISP为基础的Flavos面向对象系统[5]已采用多继承性体系，但多继承性使得系统变得极其复杂。继承路径查找即使有了可靠的算法，但其时间开销也不能不考虑。随着网络技术的发展，多继承性也许会占上风。

5. 聚束与多态性

聚束(Binding)其实并不是新概念，一个程序经编译，连接编辑成为可运行的目标码就是将执行代码聚束在一起。Pascal，C之类的准强类型语言，由于数据类型在编译时刻已确定，故运行之前即可聚束，这称之为早聚束。弱类型、无类型语言，如LISP，Smalltalk，对一个表达式释意，在编译时刻不能完成，只有在运行时刻才能确定它到底要执行的是哪段代码，故称之为晚聚束[6]。如果我们要打印不同顾客取钱凭证，会遇到有的要支票，有的要现金，有的要信用卡这几种情形。在早聚束语言中我们习惯于用Case语言，写下三种打印操作的代码，到执行时作情况判断。晚聚束其实也要做这个工作，只是由系统自动判断，找到所需对象执行，编程更自由罢了。例如，若顾客要转帐到住家附近的储蓄所，对于早聚束语言就得增加一种Case及代码，而晚聚束则是，将不同对象打印放在各自代码中，只要对该对象发出PRINT消息，就自动打印不同的凭据。虽然两者编写代码数相差不了多少，运行时间一样，但晚聚束使随意增删自如，不易出错。

早聚束在编译时刻完成，运行效率高，可在编译时刻查出聚束错误，但修改维护工作量大。晚聚束运行效率低，编译效率高，它所带来增删自如是符合近代软件可重用，可修改，可扩充要求的。随着CPU的高速发展，运行效率不是主要的。

多态性(Polymorphism)是晚聚束带来的良好特征。所谓多态即一个名字有多种语义。Ada语言已经注意到程序中一个词法元素可作多种解释。例如，运算符"+、-、×、/"既可以作整数四则运算，也可以作实数四则运算，但它的执行代码全然不同。因此，Ada有"重载(Overloading)"概念，允许一个名字或运算符符号作多种解释。不过Ada是强类型语言，编译时刻根据操作的参数类型、个数、返回值的类型，判断该操作执行哪段代码，这还不能体现多态性的长处。

面向对象系统，对象封装了操作，恰恰要利用重名的操作，让各对象自己去释意执行。而且这种多义性决不会带来混乱，这样就方便于高层设计。例如，一个学校总要上课，教育局只要发出9月1日开学的命令，各校(文科学校，理科学校，医学院，农学院)尽管上课的方式内容全然不同("执行代码"不同)但都得上课。对于早期开发，原型设计是极为有利的，因为早期不需要涉及具体的数据结构和类型，着重揭示程序的逻辑合理性。

6. 封装性

以上面向对象系统的诸多优点是靠封装手段得到"对象"而获得的。封装的目的在于将对象的用户和对象的实现者分开，用户不必知道对象行为实现的细节，只要以实现者提供的消息来访问该对象。

类似名字封装过程，面向对象语言以对象协议(Protocol)或规格说明(Specification)作为对象的外部界面，协议指明该对象所能接受的消息，在对象的内部，每个消息对应一个方法，方法实施对数据的运算。这些数据一般说来分成两部分：公有数据(继承自系统对象)和私有数据(方法中还可以设临时数据，仅为该类私有)。对数据和方法的描述是协议的实现部分或叫类体(class body)。

显式地将类对象定义和实现分开是面向

对象的一大特色。封装性本身即模块性。定义模块和实现模块分开使得维护、修改性主为改善。这也是软件技术追求的目标。

封装也是一种信息隐藏技术，用户只能见到封装界面上的信息，内部对用户是隐藏的。封装也是数据抽象技术。所谓数据抽象即如前述封装的过程与数据构成高抽象级的数据类型。

7. 面向对象程序设计语言

从以上讨论中我们对面向对象中的"对象"概念作以下补充（除第2节的三点外还有）：

(4) 对象必须是某个类的实例，类也是对象，那么它是元类(metaclass)的唯一实例。

(5) 对象有继承性，因而对不同对象有可区分的可见性。

(6) 对象的封装性要求可正式分开的外界面描述和实现描述的机制。

这样的"对象"就体现了它的模块性、数据隐藏、易维护、可修改、增量设计、可作原型开发、直接映射问题世界实体等一系列现代软件技术要求的优点。

一个语言能提供以上描述的六点，则称之为面向对象编程语言。如果能一致地以对象来看待描述的实体，则称面向对象设计。Smalltalk语言是全面体现上述六点的语言，也是当代面向对象语言的直接祖先，但它是基于对象-通讯模型，它的消息-方法通讯方式和传统语言调用-过程本质上是有区别的。显然对象通讯模型很自然地符合分布式并行程序、多机系统、网络的通讯模型。也是当前研究热门课题之一。

70年代末与Smalltalk同时研制出的Ada(80)，Modula-2(79)，CLU(77)，ALPHA-RD(81) 都提供了数据抽象和抽象数据类型的机制。Ada和Modula-2的程序包和模块满足对象的封装性，但无类无继承、CLU的簇(Cluster)是抽象数据类的概念，但无继承的概念。美国布朗大学的P.Wegner以图形形象地总结了这个分类[7](图2)：

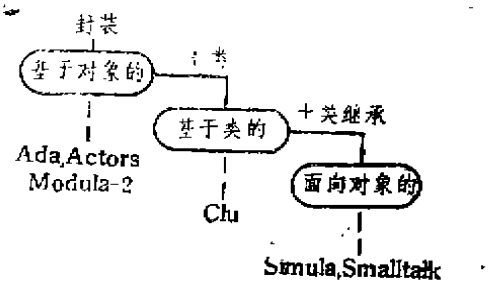

图2 面向对象语言分类

80年代以后面向对象语言的研究如雨后春笋，它们大都是基于 Smalltalk 的，可分为三类：

(1) 以传统语言结合 Smalltalk 的面向对象思想并利用通讯模型（粗俗地说，以传统语言模仿 Smalltalk)：

- 苹果公司开发的Object Pascal(85年)。
- 美国PPI公司开发的Objective-C(85年)。
- 苹果公司的 Object Assembler 用的是68000汇编(84年)。
- Kriya System开发的Neon是面向对象FORTH (84年)。
- Coral Software 开发的Objecs Logo (86年)。

(2) 传统语言采纳面向对象思想

- AT&T公司贝尔试验室85年开发C++，以C语言扩充实现部分有继承性。
- Xerox公司开发的LOOPS，它是在Intel Lisp上扩充面向对象概念，是83年推出的美国 OOP Lisp西海岸版本。
- 待专业理公司在MIT的LISP 机上开发的Flavos，具有多继承性的类对象，81年研制，85年推出。
- 也称面向对象LISP的东海岸版本。
- Xerox公司85年推出Common Loops 是面向对象的Common Lisp版本。
- 日本IBM 公司分部开发 SPOOL，1985年，是以面向对象扩充的Pro log。
- Keio大学开发 Orient 84K，1984年，是基于Prolog和Smalltalk 并能并行执行的语言。
- 西德 Kaiserslautern大学开发的Mod Pascal，1986年，是Pascal的面向对象扩充。
- DEC 公司86年推出的 Trellis/Owl，具有多继承性、强类型静态聚束，母语为类Pascal。

(3) 全面向对象语言它作基于对象设计

除Ada，Modula-2，Alphard，CLU外尚有：

- 美国Place大学White water小组为知识表示开发的Actor，用于多处理语言内类Smalltalk语言。
- Oaklisp，卡内基梅隆大学85年开发的LISP方言，在Macintosh上运行。
- Concurrent Prolog，1982年日本开发。
- Vulcan，Xerox公司86年推出的并发 Prolog的预处理程序系统。

8. 结束语

面向对象语言的基本概念最核心的两条是封装性和继承性。目前随着网络、多机系统并发程序发展，以及庞大软件系统要求强类型的安全性，其基本概念还在扩充。现在还不能肯定 Small talk是面向对象最好的语言。

当前，人工智能、数据库、程序设计语言研究有汇合之势。例如，知识的表示能不能利用类体系，数据库语言和程序语言能否合一等。面向对象是一个很有希望的汇聚点。

不管一个语言体现了多少面向对象的基本要素，用它开发的软件都要比传统语言强这一点已经得到证实。因此，90年代"对象加消息"的程序设计模式将会取代"数据结构加算法"[6]是非常可能的。

参考文献

[1] OOPSLA'86 Conference Proceedings. SIGPLAN Notices. ACM,Inc., Vol. 21, № 10/11, Oct/Nov, 1986.

[2] A. Goldberg, D. Robson, Smalltalk-80,The Language and its Implementation, Addison-Wesley, 1983.

[3] U. S. DoD, Reference Manual for the Ada Programming Language,ANSI/MIL-STD-1815A. Jan 1983. 中译，科学出版社，1985.

[4] B. Stroustrup, C++ Programming Language,Addison-Wesley, 1986. 中译，清华出版社，1988.

[5] G. E. Peterson, Tutorial, Object-Oriented Compating, Volume 1/2, Computer Society Press of IEEE, 1987.

[6] D. Thomas, What's in an Objcet, BYTE, March 1989.

[7] P. Wegner, Dimension of Objeet-Based Language Design,In Proceedings of OOPSLA'87. SIGPLAN Notices, ACM, Inc., Vol.22№10, Oct, 1987.

[8] P. Wegner, Learning the Language, BYTE, March, 1989.

全国第四次软件工程学术会议征文通知

主办单位：中国计算机学会软件专业委员会　　**承办单位**：北京大学计算机科学技术系

地点：北京　　**时间**：1991年6月

1. **会议内容**：1）学术报告；2）国外专家介绍工程研究情况；3）讨论研究我国软件工程的发展方向。

2. **征文内容**：1）软件工具和软件工程环境；2）软件规格说明技术；3）软件产品化技术；4）软件质量保证技术；5）原型技术；6）软件管理；7）软件理解技术；8）软件复用技术；9）人工智能在软件工程中的应用；10）软件工程基础研究；11）其它（新思想）。

3. **重要日期**：1）1990年11月30日前（邮戳日期）收到论文稿全文；2）1990月12月31日前（邮戳日期）发出录用及修改通知；3）1991年2月28日前（邮戳日期）收到录用稿全文；4）1991年5月上旬发出开会通知。

4. **来稿要求**：1）上述征文内容中理论研究和开发成果来公开发表者均可应征；2）论文应控制在8000字以内；3）字迹工整、清楚，用稿纸（20×20）誊写。1式2份；4）来稿无论录用与否均不退还；5）文寄"北京大学　计算机系　陈兼"（100871），信封注明"征文"。