面向对象方法学的研究

朱海滨　胡运发
（国防科技大学计算机系）

摘　要

面向对象的程序设计思想被认为是80年代的结构程序设计，从更高更广的角度，研究面向对象的方法已成为90年代的热门课题，它已不仅仅局限于程序设计领域，也逐步渗透到了软件开发、系统模拟、CAD、图形处理、数据库及知识库的组织与管理、专家系统和体系结构等计算机软件的各个方向，许多专家学者已开始从认识方法论的角度研究这一方法。以该思想为基础的程序设计语言也发展起来，并显示出强大的生命力。

本文简单介绍了几个典型的面向对象程序设计语言并对它们进行了一些比较，着重论述了面向对象的基本思想和方法学，还讨论了面向对象概念与计算机有关方向的关系。

一、引　言

自从80年代初，已有许多种新的语言相继问世，并且都声称可以支持面向对象的程序设计。其中，专以支持面向对象程序设计而开发的语言有以下几例：

Smalltalk-80系统是施乐公司经过对Smalltalk-72，Smalltalk-74和Smalltalk-76不断改进完善在1980年推出的产品。它在系统的设计中强调了对象概念的统一，并引入了类、方法、实例等概念和术语，应用了单重继承机制和动态连接。其中许多概念为其它语言所采用。

ABCL是在日本东京工学院的Akinori Yonezawa领导下用CommonLisp语言开发的，该系统的计算模型强调了对象概念的统一和并行性的开发。

POOL是由荷兰阿姆斯特丹大学的America主持开发的。它强调了并发性的开发，放弃了对继承性的要求。

Fooplog是将面向对象、函数式、逻辑式三者相结合而设计的语言。它在面向对象的概念中强调了计算的局部性。

同类语言还有Orient-84，ACTOR，LOOPS，Flavors等。

另外，还有在一些高级语言的基础上加以改进而成的，如：

Objective-C是由美国康涅迪格大学的Cox，Brad J.领导的开发小组在C语言的基础上，用类似于Smalltalk-80系统的组织和框架建成的。CBJ在该系统中提出了软插件（Software-IC）的概念，并形成了软插件库。

C++是由Bell实验室对C语言进行改进，增加了类似于Smalltalk语言中相应的对象机制开发而成的。

Common LISP Object System(CLOS)是在CommonLisp基础上支持面向对象风格特征的机制而形成的。

Concurrent-PROLOG在PROLOG语言基础上增加对象定义机制和消息发送谓词并

在此基础上实现了并行机制。

同时，还有一些自称是支持面向对象程序设计的高级语言，如：

ADA 以其所具有的包结构作为基本对象单元。它是不以继承性作为面向对象程序设计语言基本特征的。

表1 给出几个典型语言在几个要素上的比较。

基于以上的讨论，Smalltalk语言自然是面向对象语言中的最杰出代表。

Smalltalk-80系统以Smalltalk语言为核心，形成了具有多窗口友好界面的面向对象软件开发环境，它从界面，环境、工具、语言及软件可重用性等方面对软件开发工作提供了较为全面的支持。为了研究和学习，我们在VAX/VMS系统上实现了 Smalltalk-80 系统，并初步实现了 FOOPLOG 系统，从中获得了很大收益，也从实践中取得了许多宝贵经验。

表1 几个典型语言在几个要素上的比较

	Smal-ltalk-80	Objec-tive-C	Ada	C++	CLOS	ABCL/R
封装	是	是	是	是	是	是
数据抽象	是	是	是	是	是	是
继承性	有	有	无	有	有	有
连接方式	动态	双重	静态	双重	动态	动态
存贮管理	有	可能	无	无	无	无
操作符重载	可	否	可	可	可	可
软插件库	有	有	无	无	有	无
并行性	可能	无	无	无	无	有

本介着重讨论了面向对象的基本思想和方法学，以及与面向对象概念相关的问题，简单介绍了几个典型的面向对象的程序设计语言并对它进行了一些比较。

二、面向对象的思想和方法学

1.面向对象是一种程序设计方法学

面向对象方法学的研究是受到面向对象的程序设计方法研究的启发而受到重视的，面向对象方法的基本出发点是尽可能地模拟人的思维方式。

面向对象的程序设计方法起源于信息隐蔽和抽象数据类型概念，它把系统中所有资源，如，数据，模块以及系统都看成对象，每个对象把一组数据和一组过程封装在一起，使得这组过程了解对这组数据的处理，并在定义对象时可以规定外界在其上运行的权限。使用这一方法，设计人员可以依照自己的意图创建自己的对象，并将问题映射到该对象上。这一方法直接，自然，可以使设计人员把主要精力放在系统一级上，而对细节问题可以较少地关心。

1）对象是一个具有局部状态和一个操作集合的实体。

对象::=〈ID, MS, DS, MI〉

其中ID是对象的标识或称为名字；MS是对象受理的操作集合MS={OP0, OP1, OP2, …OPn−1}；DS是对象的存贮或数据结构DS={s0, s1, s2, …sn-1}；MI是对象受理的操作名称集（即消息名集，也称对外接口）MI={I0, I1, …In-1}。

一个对象就是一个自动机，DS是自动机的状态，MI表示该自动机的输入符集，MS中的每一个操作 OPi隐含对应一个状态转移函数Gi（如下图）。

2）类是对一组对象的抽象，它将该种对象所具有的共同特征（包括操作特征和存贮特征）集中起来，由该种对象所共享。在系统构成上，则形成了一个具有特定功能的模块和一种代码共享的手段。

类::=〈ID, INH, DD, OI, ITF〉

其中ID是类的标识或称为名字；INH是类的继承性描述；DD是数据结构描述；OI是操作集合的具体实现；ITF是统一的对外接

已。

从上面的定义可以看出，对象是对操作表示和数据表示两种功能的抽象。当DS→ϕ，对象→一组纯过程；当MS→ϕ时，对象→一个纯数据结构，如果真是这样，对象就成为废品一堆，无任何作用。所以，对象使得数据及相应的操作不可分，实现了数据封装的功效。

3）继承。类实现中的继承机制是一种独特的性质，它使得子类可以继承其前辈类的特征和能力。继承可以定义为一个偏序关系INH=(C, >=)，其中，C为处于继承链上的所有类。显然，继承具有传递性：IF (C2>=C1) &(C3>=C2) THEN C3>=C1，其中>=表示"继承"。

类在概念上是一种抽象机制，它抽象了一类对象的存储和操作特征；在系统实现中类是一种共享机制，它提供了一类对象共享其类的操作实现。

为了清晰起见，类可以重新定义如下(其中类1—类n属于同一类链，并且类i继承类i+1。)：

$$\text{类n::=}\langle\text{类 n 的ID}, \bigcup_{i=1}^{n}\text{类 i 本身的数据}$$

$\bigcup_{i=1}^{n}$类i本身的操作实现，$\bigcup_{i=1}^{n}$类i本身的对外接口〉

一般认为，面向对象＝数据抽象＋抽象数据类型＋继承性。

面向对象的设计方法可分为以下几个步骤：

1.问题定义：1)对象需求分析；2)对象划分；

2.对象设计与定义：1)对象的检索；2)对象的定义；

3.对象的联系：1)对象之间消息的传递；2)系统的协调。

其中，问题定义是将所要解决的问题以对象为单元进行划分，并进行整体对象的需求分析，该步骤中的对象抽象级高、粒度较大。对象的设计与定义步中的对象比较具体，粒度较小，需要进行比较详细的设计和定义。在这一步中，可以充分利用已有的对象，利用检索手段可以做到。第三步是为了达到问题的要求，将各个对象联系起来，使它们形成一个协调的整体。

2.面向对象是一种认知方法学

我们知道，客观世界是由许多不同对象组成的，每一个对象都有自己的运动规律和内部状态，不同对象间的相互作用和相互通讯构成了完整的客观世界。因此，从思维模型的角度，面向对象很自然的与客观世界相对应。

此外，从人们的认知过程来看，主要有两种方法，一种从一般到特殊的演绎方法。面向对象的方法正是这样一种认识过程，以学校为例，最初人们看到的只是"学校"这样一个词，在给其进行分类的过程中就可以不断理解这一词的含义。进一步知道学校有大学，中专，中学和小学之分；再进一步又知道大学又分综合性大学，理，工，农，医，文科大学等；每一科又分为不同专业，专业又分为不同方向，…

在分类到一定程度之后，以面向对象的方法，就是对已经分好的各类对象进行状态描述和功能定义，以明确这一类对象所能完成的工作，其实这也是一种分类，大学由部，系组成，各系又由多个教研室组成等等。

最后一步就是怎样让这一类对象运转起来，这就是使各类对象建立联系，使用继承和类比方法进行状态转换，完成它们应有的功能，对于学校这一系统，就是建立各教学单位与各教学保障单位、教师与学生的联系，从而使整个教学系统正常运转起来。

另一种是从特殊到一般的归纳方法。我们今天看到一条黄狗，它是一个对象，明天又看到一条白狗，它也是一个对象，这两个对象除了在颜色上不同外，其它狗的特征完

全一样，这样，我们便可以构造一个类——"狗"，其中描述了狗的所有共同特征，比如，会叫，具有犬齿，嗅觉灵敏，具有颜色，忠实等等，而黄狗和白狗都是这个类的实例。因此，面向对象很适合这种认识方式的组织。

面向对象既然提供了从一般到特殊的演绎手段（如继承等），又提供了从特殊到一般的归纳形式（如类等），从而说明它是一个很好的认知方法。只不过目前以面向对象为基础的归纳方法的研究尚未展开并引起足够的重视。

3.面向对象方法与其它方法的比较

从横向来看，当前程序设计领域中研究的重点，根据对计算过程的不同认识可分为三种：

第一，函数式程序设计（FP）将计算过程看作为函数作用过程，即将某一系列函数作用于输入得到输出的过程，其中强调等值替换，LP则无法做到这一点；

第二，逻辑程序设计（LP）将计算过程看作为推演过程，即将具有初始状态的输入在一系列条件的约束下，采用推理算法和搜索手段进行匹配、演算的过程，有利于启发式；

第三，面向对象的程序设计（OOP）则将计算过程看作为分类过程加状态变换过程，即将系统逐步划分为相互关联的多个对象并建立这些对象的联系，以引发状态变换，最终完成计算。

其中，面向对象方法模拟了人类认识问题的较高，较广层次的过程，即分类过程，属于战略性方法；函数和逻辑方法则更适合于模拟人的逻辑思维，处于人类认识问题的较深层次过程，属于战术性方法，它们的结合才是最完美的。

纵向来看，面向对象方法被认为是80年代的结构程序设计，那么，它们之间有什么区别呢？下面从几个角度进行讨论：

结构化程序设计强调了功能抽象和模块性，它将解决问题的过程看作是一个处理过程；而面向对象的程序设计则综合了功能抽象和数据抽象，它将解决问题看作是一个分类演绎过程。

1）模块与对象　结构化设计中模块是对功能的抽象，每个模块都是一个处理单位，它有输入和输出。而对象是包括数据和操作的整体，是对数据和功能的抽象和统一。可以说，对象包含了模块。

2）过程调用与消息传递　在结构化程序设计中，过程为一独立实体，显式地为它的使用者所见；而在面向对象的程序设计中，方法是隶属于对象的，它不是独立存在的实体，而是对象的功能体现。消息传递机制很自然地与分布式并行程序、多机系统、及网络通讯等模型取得了一致。在结构化设计中同一实参的过程调用，其结果是相同的，如P(x，y) 是一个求2*x+3*y的过程，那么P(4，5) 的结果在每次调用都是23；而面向对象中的消息传递则不同，同一消息的多次发送可能产生不同结果，如go aPoint这一消息是指出一个点画一条到点aPoint的线。但go 50@100这条消息发送给点50@50这样一个对象时，则是画一条竖线，而go50@100这条消息发给点10@100，其结果是画一条横线。其中，点的表示X@y，x表横座标，y表纵座标。

3）类型与类　类型与类都是对数据和操作的抽象，即定义了一组具有共同特征的数据和可以作用于其上的一组操作。但是类引入了继承性质，实现了可扩充性。

4）静态连接与动态连接　从程序设计发展来看，用户对灵活性和方便性的要求不断增强，所以动态连接代替静态连接是个必然趋势，计算机硬件速度的提高弥补了动态连接的低效性，为此提供了基础。显然，面向对象在这一方面便比结构化设计占了优势。

4.面向对象的基本特征

基于以上对面向对象方法的讨论，面向对象具有以下特征：

〈1〉模块性：一个对象是可以独立存在的实体，其内部状态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〉封装功能：对象将自己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速度。

〈3〉继承性：系统的处理能力可以通过对象的继承性实现共享。

〈4〉动态连接：对象的功能执行是在消息传送时确定的，这样可以提高程序设计的灵活性。

〈5〉易维护性：对象实现了抽象和封装，使其中可能的错误局部于本身，不会传播，易于检错和修改。

〈6〉增量型设计：表现在继承性可使系统的功能不断扩充，而不影响基于其上软件的运行。

三、面向对象的相关问题

1.面向对象与数据库

1）数据模型　对象是数据和操作的统一体，不仅拥有关系模型的一切优点，而且具有关系模型所没有的丰富操作功能，对象的层次性也进一步保证了数据的无冗余性和一致性。

2)持续性　持续性是用于确定对象保持时间的一种性质。数据库所存数据的存活期很长。而对象可看作为一种具有特定性质的并且存活期长的对象，它可以为许多用户共享。而每个对象都有自己唯一的标识——名字，它独立于对象选择中所用的关键字并持续存在于系统中。

3）版本控制　版本控制是数据库的特殊要求。在面向对象的数据库中，版本控制的实现与对对象的认识有着密切的关系。

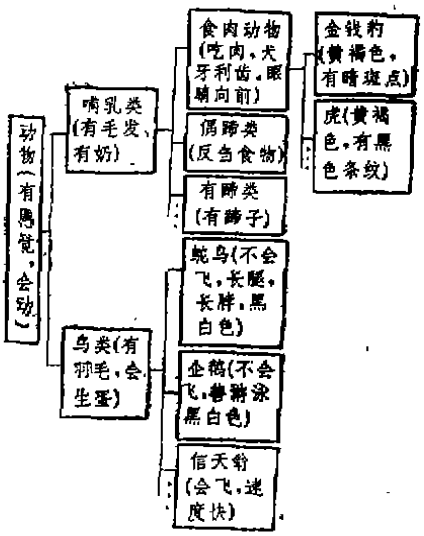
一种认识是，对象是不可变的，任何外界对对象的刺激，都将产生新的对象。在这种认识下，给每个对象增加两个属性“变换成”和“变换自”，就可以清晰地表示对象的版本历史。

另一种认识是，对象是可变的，外界的刺激将导致对象内部状态的改变，是否产生新的对象只由对象的操作来决定。基于这种认识，实现版本控制，则需要对对象的标识进行改造，即将对象的名字增加一个版本标识，这样对象的属性不必改变即可实现对象版本历史记录。

4）查询语言　这一特征也是数据库的特殊要求。查询就是要在数据库中找到某些条件限制下的一组对象。继承机制为此提供了良好的基础。依面向对象思想组织的数据库中实现查询语言就是树或格的遍历问题。比如，关系数据库中要查找会飞的鸟，可以用*select Bird from Animal where fun=flying*。而在OODB中，完成这一功能只需要延长动物类链，查找到其功能属性中包括“会飞”的所有鸟的子类或实例即可。在这种要求下，OODB中的类不仅要保持子类的信息，还要保持其实例的信息。

2.面向对象与知识表示

1）分类知识表示　面向对象中继承机制直接支持了分类知识的表示。比如，动物世界可以由以下分类图示表征。



其中，每个结点表一类动物，括号内表该类对象的属性。在上述各类组织完后，再

加上必要的方法，就可以形成一个简易知识库了。

2）启发式知识表示　面向对象方法中继承机制很自然地表示了分类知识，但对于启发式知识的表示则是比较困难的，虽然可以应用消息来模拟规则的表示，但非确定性知识的用法是难以表示的，并且目前的面向对象系统是没有内含的推理机制的，所以要以面向对象的方法组织知识库就要增加启发式知识表示机制。这也是为什么要结合逻辑程序设计和面向对象程序设计的原因之一。

3.面向对象与操作系统

面向对象与操作系统之间存在两方面关系。一方面是以面向对象的思想如何组织操作系统，另一方面是操作系统如何支持面向对象系统的运行。

以面向对象的思想组织操作系统有以下几个优点：

1）OS的设计需要管理多方面的资源信息，如文件，打印机，处理机等，用对象来描述更为自然。

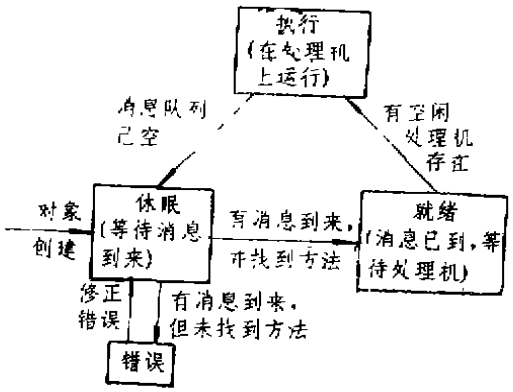
2）设计OS有许多工作要做。如命名，同步，保护，管理等，这些工作由于引入对象的概念而变得易于实现，便于维护。

3）对象对于网络处理十分得当，每个结点可以描述成为多个对象的集合，这样便可以提供大量不同特性的结点，使网络更加丰富和协调。

4）对象模型有效地维护了有关对象信息的数据库。

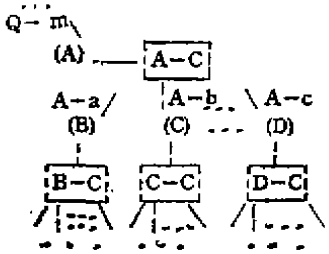
另一方面，OS的设计要考虑到支持面向对象系统的运行必须提供高效存储管理，处理机调度，对象状态管理和消息传递机制。

根据执行情况的不同，一个对象所处的基本状态有休眠，就绪和执行三种状态，另外，还有可能出错，其转换如下所示。



消息的处理和回答的实际处理应由应用程序确定，操作系统所作的工作主要就是对象运行状态和消息队列的管理。

最大限度地发挥各个处理单元的能力，是支持面向对象程序系统的难点和重点，操作系统也就必须负起这一责任，对象到处理单元的适当映射显得非常重要，特别是在分布主存的体系结构中，对象映射的适当与否将直接影响到整个系统的高效执行，比如，对象在处理过程中应最大程度地在所处处理单元对应的存贮器中查找方法，否则，不同处理单元之间的消息传递将会大大影响系统的执行效率。在对象处理消息的过程中，不断启动新的对象是很经常的，这样就使得许多不相关的对象可以同时运行，并且，活跃对象的数目大约是按指数增长的（如下图）。这也就是为什么要设置快速语景（语景是面向对象系统中对象的工作状态，语景是真正在处理单元起作用的实体）切换部件的原因，这也要求操作系统处理单元的分配上给出好的算法。



其中，（）表示对象；| |表示对象对应的语景；X-y表示X对象所发的y消息。

4.面向对象与体系结构

从体系结构的角度来看，面向对象具有以下两个特点；

1）固有并发性：对象是自治的，多个对

象可以同时处理自已接受到的消息，相关性较少；

2)局部存贮与分布式计算：即，每个对象通过数据抽象和数据隐蔽将其内容和状态置于自身独立的存贮结构中，并且，对象的处理也是自治的，整个系统的运算和处理是分布式的。

以面向对象的思想组织系统或以支持面向对象系统的运行都是一致的。以每个结点代表一个对象或一个对象集合，加上支持消息处理的专用硬件机制，则是最基本的设想。

要支持对象功能的并发执行，必须建立多个处理单元的硬件环境。可以设想采用主辅机结构，主机是通用计算机，辅机由多个具有专门硬件的处理单元组成，并配以适当的互联网络。

采用紧耦合（共享主存）结构的多处理机系统可以减少消息的发送量，具有大量紧耦合对象的软件系统在这种结构上运行可以减少通讯量，因为许多对象的启动都是动态的无法确定其分布情况，这种结构的弱点是Cache与主存不一致性的处理要有一些开销。

采用松耦合(分布主存)结构的多处理机系统更接近于面向对象程序设计的概念。这种结构适合于对象分布和耦合情况比较均匀的软件系统，这种系统的主要工作，分别由各个处理机完成，相互之间的消息传递较少，这种结构对于对象松紧耦合程度平均的系统更为适宜，因为，紧耦合对象可在相应局部存贮器中进行消息传递，路径短，开销小，松耦合的对象又可以占用不同处理机，使多个处理机都能处于忙碌状态。

神经网络计算机的体系结构与面向对象的体系结构具有类似的思想并能够相互支持，促进，一个神经元就是一个小粒度的对象。连接机制与面向对象也有着天然的联系，一次连接可以看作为一次消息发送。

四、结束语

以上重点从方法给出了面向对象的介绍和描述，力图给出一个较全面的认识，并想借此文能够起到抛砖引玉的作用，引起大家在广义上研究面向对象方法的兴趣。

虽然，目前对于面向对象的研究还缺乏坚实的理论基础和充分的形式化定义，另外各人对于面向对象的看法也不尽相同。但无论如何，面向对象的思想已经开始受到了广泛的重视。同时丰富而卓有成效的实践也必将推进面向对象的研究，使之向着更深入和更广泛的方向发展。

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. Brad J. Cox, Object-Oriented Programming--An Evolutionary Approach, Addison-Wesley, Reading, Mass., 1986.
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 Programming 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. P. Wegner, "Learning the Language", BYTE, March, 1989, p245-253.

Stephen S. Yau, Jeffery P. Tsai, 软件设计技术之概观，计算机科学，4，1987，pp10—18.

计算机科学1990№.5

面向对象并发程序设计导引

Akinori Yonezawa和Mario Tokoro

面向对象的并发程序设计是一种程序设计方法学，也是一种设计方法学。在这种程序设计中，待建立的系统被模拟为一个称之为对象的可并发执行程序模块的集合，它们之间的相互作用靠发送消息来进行。本文系《面向对象的并发程序设计》文集的卷头文章，虽然重点是介绍有关这一方法学的当前工作，但有一定的普遍意义。

背景

面向对象的并发程序设计这一技术的研究出于我们需要设计和建造强有力的，而且是灵活的计算机软件系统，以满足社会需求的不断增长。随着信息化社会的发展，计算机系统应能解决更复杂的问题并提供更高级的服务。尽管以适度开销，生产更快更大的机器的技术可能继续向前发展，但直接使用这种机器不会产生能满足我们所要求的计算机系统。我们需要的是并行工作，使得我们能充分利用由多台计算机组成的大量计算结点并使它们合作运行。

并行性

从概念上来说，并行工作是吸引人的，但有效地利用并行性并非易事。在计算机科学传统分支，如操作系统设计和分布式数据库中，已经发展了控制并发性的许多有用的技术。然而，这些技术对建立所要求的系统是不够的，因为待构造的这些系统的构件之间需要更广泛的各种相互作用和更高的并发程度。

实现并行性的中心问题是什么活动和谁的活动应当并行执行，以及这些并发活动应当怎样相互作用。在设计一个体现并行性的软件系统中，这些问题归结为应如何将该系统分解为可并行活动的构件的问题，以及如何划分构件的功能的问题。为了保持系统的透明性，分解应该是自然的和模块化的。那么，每个构件应当以什么样的风格或形式表示呢？正如本卷研究报告中提出并作简要讨论的一样，“对象”的概念为我们提供了一种非常有希望的形式。

对象

“对象”这一术语几乎独立地在计算机科学的各个领域中出现过。早在70年代，差不多同时涉及到了一些表面上看来不同，但相互

9. 朱海滨等，“Smalltalk-80虚拟机在VAX/VMS上的实现”，小型微型计算机系统(10)，1989。
10. 朱海滨，“基于Smalltalk语言的面向对象程序设计”，微小型计算机开发与应用（6），1989。
11. 朱海滨、陈火旺，“从Smalltalk语言的结构看‘软插件’的形成”，计算机科学（3），1990。
12. 朱海滨，“Smalltalk语言虚拟象的设计与实现”，第二届系统软件研讨会，哈尔滨，7，1989。
13. 朱海滨、徐锡山，“面向对象的程序设计——方法、语言及实现”，计算机工程与科学(3)，1990。
14. 朱海滨、胡守仁，“面向对象计算机研究中的几个问题”，计算机工程与科学(3)，1990。