1983年全美计算机会议论文集

中国科学院成都计算机应用研究所情报室译

AFIPS

1983年全美计算机会议论文集

上卷

中国科学院[illegible]计算机应用研究所

软 件 工 程

原　　序

DON B. MEDLEY

为了给1983年全美计算机会议制定一个高质量的技术程序，一个干练的计算机专家小组已经进行了长时期艰苦细致的工作。这本论文集代表了为使会议程序富有教益而筹备的许多献礼的记录。在未来的岁月中，这本论文集将为计算机专业人员提供宝贵的参考资料。

除了这本论文集所包括的论文以外，在1983年的全美计算机会议上还有许多小组会和报告会，构成了完整的全美计算机会议富有教益的程序。我希望你们抽空参加一个或几个这些有益的集会，而这本论文集在今后的岁月中将是你的一个有用的信息源泉。

前　言

ALLEN N. SMITH

“突出的信息社会：计算机，通信和人”是1983年全美计算机会议的议题。在编印这本论文集时，这个议题对这次会议来说显然是非常适宜的。会议程序中的多次集会与论文，解说和有关其它途径的想法在几条不同途径上密切结合。论文集中的不少论文可能在几条不同途径上发表。显然，现在的计算机领域是一个多学科的领域。因之，论文集着重在办公系统，个人计算机，远程通信和人的因素。尤为重要的是对所有这些学科进行综合，从而理解全部领域的必要性目前正日益增长。我们希望1983年的论文集提出这样的见解。

基于这个议题，1983年的论文集分为以下九个主要的范畴：

1．软件工程——为未来的系统开发提出新的想法和引导。

2．管理／教育——各种管理和教育问题的广泛探讨，特别是系统维护方面日益增多的问题。

3．数据库／分布式系统——在数据库管理用的软件和硬件方面的更新趋向。

4．人和社会的问题——广泛包括计算机对社会，机构和个人所产生的影响问题。

5．办公室自动化——在办公室自动化的多方面进行更新，包括个人计算机对这个领域日益增长的影响。

6．决策系统——在这个范畴内，包括行政和专业支持的重要性日益增长所产生的新领域。

7．硬件——对新型计算机硬件和体系结构的新趋向和研究工作进行探讨。

8．远程通信／应用——在最新的远程通信研究工作中作出一系列更新，并在应用和使用技术上进行一系列简易的革新。

9．个人计算机——对迅猛增长的微型机（包括个人的和商业的用途）领域广泛地进行评述。

先驱者日的内容是Howard Aiken和Harvard计算技术实验室，回顾一下计算机领域中早期的开发工作。

与1982年的程序一样，我们将集会的数目减少到84次，而在过去的程序中，集会次数超过100次。这样作就使我们能够将程序和论文集集中到最重要的领域上，从而提高质量。从投交来的大量论文中我们选出了80多篇论文；我们相信所选出的一篇一篇论文在它们的领域中都是高质量的。我们不得不取消许多很好的课题和谢绝很多很好的论文。会议程序包括论文集中论文页码的标号，便于查阅。在论文集中没有编印小组讨论的总结；但在论文集中包

---

* 先驱者日是会议的活动日程之一——译注

括有各个领域的简要总结。

1983年全美计算机会议程序的制定需要很多人专心致志的努力：程序委员会的成员，集会主持人和领导人，小组长和解说员，技术论文的作者，以及帮助我们选定这本论文集中所列入的论文的审查人。此外，在编印论文集的工作中，AFIPS 的工作人员对我们的帮助很大。以Jeff Young为首的委员会工作人员，与Carrie Borgen及Georgia Marinlli的无偿援助对本书的编印工作贡献甚大，深致谢意。对所有这些人，特别是对程序委员会，我个人都表示感谢。我们衷诚希望这个程序对来宾能提供有用而且引人入胜的知识。

目　　录

软 件 工 程

管理/教育

数据库/分布式系统

人类和社会的问题

办公室自动化

少写代码——可实现的理想

NAOMI LEE BLOOM

摘　要:

我们正在被不满意的用户期望的海洋所淹没。应用开发要求的积压继续在增长，有时竟到了令人吃惊的程度。对于这种积压已经进行过很多讨论，但却几乎没有什么减少。一种较通用的减少积压方法是力求提高我们缺少的技术资源（程序员、系统分析员等等）的工作效率。大大减少为了满足用户应用需要必须编写的新代码数量，这可能是一种较有前途的应付这种需要的方法。无须深入理解就能确信：在其它条件相同时，为达到规定水平的系统支持而编写的代码愈少，风险、费用和经过的时间就愈少，挫折也一定能为机构所接受。本文简要地综述一些一般的和一些不太显著的可用技术。最有前途的技术中的两种，基础软件和可改写的应用软件包，在另外的论文中分别进行了更详尽的叙述。

引　论

我们正在被不满意的用户期望的海洋所淹没。应用开发要求的积压继续在增长，有的竟到了令人吃惊的程度。对于这种积压已经进行过很多讨论，但却几乎没有什么减少。而且还有看不见的积压，Martin 称为用户没有讲出的（和也许还未想到的）要求，使这个问题确实不可能减小。

一种较通用的减少这种应用积压的方法是力求提高我们缺少的技术资源（程序员、系统分析员等等）的工作效率。提高效率的技术，诸如结构程序设计、结构化分析、回归检验和交互式程序设计已被广泛采用，但是积压仍在增加。很明显，即使所缺少的技术资源的工作效率有量的飞跃也不会消除这种积压。

大大减少为了满足用户应用需要必须编写的新代码数量，这可能是一种较有前途的应付这种需要的方法。用看来是起源于 IBM 的话来说，这些“可能应用的技术”是用来减少编写新代码的数量而不是仅仅加快新代码的生产。无须深入理解就能确信：在其它条件相同时，为达到规定水平的系统支持而编写的代码愈少，风险、费用和经过的时间就愈少，挫折也一定能为机构所接受。

然而，重要的是要注意到其它事物常常并不相同。本文叙述的多数方法都用增加计算资源消耗量去换取为达到某种水平的用户支持所需人力资源的减少。因为在硬件成本继续下降和随之而来的价格性能比不断改善的同时，有才能的分析员、程序员和有关的计算机专业人才变得更加缺乏和更加昂贵，

在商业上采取合理折衷方案是明显地用增加硬件资源消耗量换取研制工作所需人——时和用户时间。这种折衷方案当然不得影响用户需要的满足，并且对每种应用都必须仔细估计，使与各种软件包和工具相关的系统开销不会使项目小组感到意外。

本文简要地综述一些一般的和一些不太显著的可用技术。最有前途的技术中的两种，基础软件和可改写的应用软件包分别在Curtis及Woodward和Di Giammarino的文章中详细叙述。如果使用得当，这里所提供的方法不仅将减少由任何一个机构编写的新代码数量，而且会减少新代码的总量。然而，即使这些可能应用的技术得到充分应用，仍然不得不编写一些新的代码，而且写这些代码应该是高效率和有规则地进行。在本文及Curtis和Woodward与DiGiammarino的那些文章中，焦点主要集中在传统的商业应用上，这些可用技术同样可用于开发科学应用，面向系统的应用或个人的应用。

可用技术的系列（或层次）

寻求少写代码的方法并无特别难以理解之处。你可以做下列任何工作：

1. 使用户确信不要求（或不需要）新的应用。
2. 重新使用旧代码——你自己的或别人的。
3. 用简单工具（记注如何操纵）使你编写的任何代码的工作价值倍增。
4. 让别的人，也许是你的用户，为你编写代码。

使应用能够成功的关键是把这些非常简单的准则运用到你的系统开发生命周期方法中。不应该开始应用开发或甚至软件包安装项目，除非新的应用确实需要。在生命周期的每一阶段，你必须问自己要研制新代码有那些可供选择的方法。因此，在某种意义上讲，可用技术平行于应用开发生命周期。

在常被称为商业系统或战略系统规划阶段的最初阶段，你必须提问基本的问题，即这种应用是否真的值得做。这个步骤通过后，你应当提问下列类型的问题：

1. 这种应用从前是否开发过？如果是，有哪些旧代码你能重新使用？
2. 这种应用是否有助于使用简单工具？是别人研制的工具或是你自己研制的工具？
3. 这种应用是否有助于终端用户编写代码？终端用户编写代码是很多数据操作和分析应用的特点。

通过在系统开发生命周期中适当时候提问这些问题，你就可以利用很多使新编代码减至最少的技术。本文的其余部份按这些技术在生命周期中出现的先后顺序进行探讨。

不开发不必要的应用

最显著地解决如何少写代码问题的办法是删去除基本的（换句话说是正确的）应用以外的一切需求积压。战略系统规划（也称商业系统规划）是这样一个过程，通过它，一个机构可以辨别出并优先安排它的主要系统开发目标。明确地调整应用开发的优先顺序与机构的商业战略相一致，我们就朝着减少新代码迈出了关键的第一步。

虽然在文献中描述的战略系统规划有很多使用方式，但是IBM在他们的商业系统规划方法中确定的目标是有代表性的：

1. 给管理部门提供建立总信息系统的优先顺序而不管局部利益的正规目标方法。

2．确保缺少的开发资源用于生命长的系统，从而保护系统的投资额，因为这些系统是以一般不受机构改动影响的商业过程为基础的。
3．使数据处理资源的管理达到对商业目标最实际有效的支持。
4．增加执行者的信心，相信可以产生出高利润的主要信息系统。
5．通过提供对用户需求和优先反应灵敏的系统改善信息系统部门与用户的关系。
6．把数据视为应当进行规划、管理和控制的公用资源，以便每个用户都能有效地使用它。

我们应确保只开发那些与机构的关系和利益已经严格审查过的应用。这样，我们就实现了使不支持的应用需求积压减至最少的第一个突破。再说一遍，如果你不开发不必要的应用，就不要指派你去写（和维护！）没有价值的代码。

重新使用旧代码——自己的或别人的

当一个应用课题被证明是应该解决时，有几种办法可能做到既研制它而又无须写代码或者只写很少量的（正是所希望的）简单代码。应用软件包已经用了近30年，它们足以支持很多日常事务（和系统，如分类）功能。另外，你的很多事务功能（如像特殊数据元素的编辑程序）可能在你的机构中已经编写了很多次。因此，在判定这一个应用课题是否如此独特以致无法使用任何现有代码之前——很多内部分析员和用户的共同看法——应首先考虑软件包和可重新使用的内部代码可否使用。

当前通用的商品应用软件可以分为三大类：

1．旧式的软体包，它可以完成一组意义明确的功能，但安装它的选择可能极少。
2．现代的软件包，它可以根据很多表控制的，用户定义的和安装规定的选择完成一组意义明确的功能。
3．可改写的软件包，它可以根据很多表控制的、用户定义的和安装规定的选择完成一组适应性很强的功能。

旧式软件包

最初，应用软件包实际上是定制的软件，它只不过作为补充，由研制者挑选来与别的用户共享。早期的软件商常常销售的基本定货系统只带有最少的文件和安装支持。安装这样的软件包要求买方修改代码，甚至要支持最明显的安装规定要求，例如，改变报表标题使它包含买方公司的名称。

旧式软件包（和很多目前正在销售的软件包）的买主得到一些明显的好处：借助简短的介绍就能够得到和安装经过调试的代码，在对源代码进行最小修改后这些代码就可以执行一组意义明确的功能，他购买软件包所付的价格远比为同样目的专门订制而进行研制的费用要低得多。不用说，上面强调的形容词要看买主个人如何评价。但是从理论上讲，购买旧式软件包所承担的风险、付出的费用和经过的时间（和很可能要受到的挫折）都比从头研制该应用要少。

这些理论上的好处在实际中并不经常都能实现。在采用旧式软件包时，每个用户确定的要求，从报表标题和格式到公用算法的改变都会导致修改外来的（最好的情况）源代码。或者经常要修改的是那些难以理解的和没有编制文件的源代码。由此看来，虽然旧式软件包仍然是一种合适的少写代码方法，但它们的不灵活性可能会给应用带来挫折。

现代的软件包

使用现代的（即科学的）软件设计方法，同时注意到即使是最灵活软件的买主也会有一些特殊要求，这就导致产生一种新型的软件包。概括地说，商业化软件产品即现代软件包（我的术语）有下列特点：

1. 编成完善文件的源代码在结构上提供了风险小的用户（子程序）出口，这就是说，在源代码的特定位置处可以插入用户写的子程序而无须中断程序流或取消卖方的被保证人。
2. 参照表，它们从源代码中调出哪些频繁地专门定制的功能，如像报表标题，有时也包括格式，信息代码文字和严重级别，数据元素名、字段长度、数据类型和编辑规则，包括有效值和代码转换参照表的指示字；参数值，例如过程进度日期、当前扣留的税金百分比、航空公司订票超额百分数，计算算法在高级软件包中不仅参数值而且操作员和计算基础都是表控制的；编码结构，例如账目或机构结构的表格也是表控制的。
3. 正规的安装过程，包括转换程序、作业流和其它减少代码的辅助手段的样本。

像旧式软件包一样，购买和使用现代应用软件包一般可以减少满足系统支持需要的费用、风险、经过时间和个人的挫折。然而，为了灵活性总得付出代价。完善的参照表比修改源代码的风险小得多，但这种方法要求大量的装入和维护工作。加之，用户常常可能被束缚到负责装入和维护大多数这类表格的任务上。

更重要的是从包含的价值、风险和经过时间看，要有效利用选择的软件包意味着必须有人（通常花费很大）分析、编制文件、介绍、估价和决定（正是希望的）各个符合要求的选择。虽然如此，但现代应用软件包对于不研制新代码就满足机构的专门需求毕竟大有帮助。

在进入软件包的新领域之前还有一点要注意。正如早先提到的，要使用通用软件时通常会有硬件资源消耗量增大的障碍。现代软件包支持表控制处理超过硬编码处理，在这方面它比旧式软件包有更大障碍。

可改写软件包

软件包的一个最有意义的新发展是趋向于建立有关的模块组，这些模块可以重新组合以适合各种不同的应用要求。为支持信用卡的收付款已研制出一个这种软件包（CACS）。信用卡收付款是一般应用的一个特殊例子，包括事件跟踪、调度和状态处理功能，所研制的这种软件可使这些一般功能自动化。把包括过程控制表在内的很有作用的参照表与能以各种不同方式组合的程序模块配合使用，对源代码作最小修改CACS就能用于支持多种用户需求。Woodward和DiGiammarino的论文更详细地叙述了CACS和可改写软件的概念。

用旧代码解决部份问题

早期，对数学子程序和统计程序的访问增强了很多编译程序的功能。在现代的系统中，灵活的数据代码词典常常从公用或共享子程序中驱动数据元素编辑。的确，很多数据处理公司都已研究出一些与COPYLIB大致等效的标准源语言软件部件，这些部件在不同的应用中能够以最小的风险、成本等等进行再生产。当我们讨论使用简单工具影响任何所写代码的价值时，要进一步发展的一个

观点是设计工作必须明显地把焦点集中在标识这样的公用处理上，即它们可以只编写一次程序，而不需要在多道程序、系统或装置中重新编写。

为了充分利用现有代码（或是辩别为最初开发而进行的通用处理），生命周期方法必须强调在设计的各个水平回答下列问题：

1．以前我们曾使这种功能自动化吗？甚至一个较小的（如数据编辑）功能可能曾经一度作为一般的例行程序编程过，其成本要比让每个程序员都编自己的程序要低得多。如果你很机灵，已经使一个单独的日期例行程序编入你的系统中，那末至少到2000年你的工作将大大简化。最重要的是要以这种方式把每个过程作为机构标准软件库的一个可能候选者来看待。
2．我们是否还需要使这种功能再次自动化？日期编辑、把机构代码翻译成它们正确名字、报表标题和很多其它公用功能几乎在每个商业应用中都有。最初以稍大的成本照一般方式编写这些功能，以后就可永远使用它们。

除非在研制生命周期的各个阶段明显地提出标准软件（可再用代码）问题，现在和将来很多少写代码的机会都会失掉。

使用简单工具

有两种通用的方法增加你编写的代码的价值：

1．扩展软件。你用这种软件工具的命令语言写简单代码（正是所希望之点），该软件以你写的代码作为输入并从它（通过翻译、编译、汇编或者其它的某种扩展技术）产生很大的功能。
2．保守技术。它是一组正规的设计技术，它们在应用中寻找公用的功能元素，这样，只要研制这些公用功能的个别实现，就能在整个应用中使用。

重复使用数据例程是非常简单的保守情况。对刚才提到的两种方法，在本章中我们还要仔细研究一些更复杂的例子。

扩展软件

当你为调动IBM的各种操作系统的机能而写JCL时，你正在使用扩展软件使必须写的代码最少。我最早在IBM1401计算机上以机器语言编写的程序没有这种扩展部份，并且我们编写我们自己的磁带读入和打印机输出。现在在你的应用中，每个系统实用程序的使用，亦如调用COBOL内部 SORT，只变动很少几个实用程序命令就可完成相当多的工作。

因之，扩充软件的范围包括从老的和熟悉的一直到新的和仍在研制的软件：

1．实用程序，它们提供系统或内务功能。
2．报表书写程序和查询语言，包括绘图软件包。
3．数据库管理系统，通过它，在应用程序中你使用简单的命令就能调用强有力的数据处理、编辑、存贮和访问能力。
4．屏幕生成程序。
5．数据管理和分析工具，例如，SPSS和SAS。
6．应用生成程序。
7．非常高级的语言。

这些工具之间的界线并不明确，且其中很多都可由非技术人员使用，从而在根本上改变应用开发的职责。所有这些工具都同样有希望提供复杂软件，以便用简单命令调动很强的功能，并且正在提供很多工具。

然而，扩展技术的使用也有一桩麻烦事。这就是我们现在正淹没在命令语言、专用语法以及类似英语的易学语言的大海之中。甚至对如何确定命令的界限还没有一致意见！直到值得重视的标准化出现以前，甚至只利用少量的这些工具也将会使机构承担严重的训练负担。并且很多专业的程序员和用户将反对使用这些工具，因为他们完全有理由看出掌握这些工具的费用太高。

保守技术

有理解力的分析员和设计人员总是在他们的应用技术说明中识别出公用的功能，但是这种作的过程很不正规。在很多商业应用上，有相当大的一组通用功能，这有助于使用通用软件。在美国管理系统公司，我们已经把一个相当正规的寻找这些公用系统元素的过程结合进我们的生命周期方法中。

决定建立以公用软件模块为基础的应用必须在设计过程中尽早做出，以便后面的工作都能做到有的放矢。我们称由此产生的为应用的其余部份提供公用服务的软件为基础软件。研制大型应用系统的基础软件方法在Gary Curtis的参考论文中有详细叙述。

让别人写代码！

终端用户计算并不是一种新设想。在计算机发展的初期，设计程序是科学家、工程师和数学家的附带工作，他们企图使用计算机这个庞然大物去计算弹道导弹的轨迹和为其它同样令人生畏的问题研制软件。在我当程序员的初期，会计们还在研制最初的自动化的工资单、通用账目和银行及保险系统。专业程序设计的年令还不到20年，那么为什么我们现在要把终端用户计算看作是一种现代化发展呢？

一个原因是，直到现在为止，任何接近计算机的人都被迫学习计算机专用语言——这是很大的人力损失。如果我们相信对各种终端用户计算机工具的广告，专职程序员可能很快就只集中到磁心生产系统和工具研制上，给用户留下大多数数据析取和分析（MIS）系统的开发。可是，这种前景还没有到来。

本文叙述的很多简单工具可以由非技术人员在接受一些训练后使用，并且信息中心的增长更表明便利用户的工具是有效的。例如，RAMIS Ⅱ或FOCUS这样的第四代语言被宣传成了用简单命令发展所有应用的有力工具。个人计算机的激增证明用户倾向于像Vi siCalc这样的工具。显然，如果用户能直接把他未讲的（或讲得不清楚的）信息需求解释给工作系统，他就不会让DP职员再仗势欺人了。

结论

要是没有程序，那怕是软件、软件硬件相结合的固件，或者作为硬件一部份的程序，计算机都没有价值。现在人们仍要写程序，并且人力是昂贵的、无法预见的和脆弱的。如果只是为了多卖计算机，硬件商人应当会欢迎（支持、可能还会提出）这样的研制方法，这种方法开发使用更多计算资源的新应用，而本身又为解决人力缺乏问题需要使用更多的计算机资源。对于软件商来说，由于他们研制很多软件和工具并且普遍垄断着真正超级的专业程序员市场，他们一定会喜欢本文描述的技术。用户和DP管理部门也都乐于少写代码。既然如此，为什么应用积压还在继续增长呢？

1．内部程序员宁愿写程序（不是指设计整个系统）而不愿为现代软件包装入各种表格或安排报表编写程序。或许我们需要一种新的DP辅助人员或辅助专业人员，他们把使用本文所说明的工具看作合意的工作。

2．语法或句法没有标准化，现在可用工具无论在那儿使用都会产生巴比塔效应。（意为语言太多造成混乱——译注）

3．很多用户使用终端程序块，这限制了他们使用计算工具的能力，其它各种不便就不提了。

4．计算资源是变得不算昂贵了，但毕竟不是免费的。要获得总是在大量增加的资源对于机构来说是比未满足应用需要的（可能）价值更显著的消耗。

时间无疑是站在本文叙述的方法一边，但是我不会解雇我的COBOL程序员，也不会断言所有的用户需要能由他们的新应用所满足！如同所有其它事情一样，这些新技术与传统的应用研制战略的均衡配合将产生最好的结果。

杨德明　译

吴浣尘　校

基础软件：开发大型应用系统的一个重大改进方法

GARY A. CURTIS

摘　要

美国管理系统(AMS)解决大型应用系统技术结构的方法是根据我们称之为**基础软件**的概念，这是一个由标准程序包和为开发及运行应用软件提供公共服务的用户模块组成的综合环境。该环境为应用软件提供了一个标准化的、结构化的和简化了的外部世界视图。基础软件的应用戏剧性地改善了大型系统开发，运行和维护的经济性能并减少了开发这类系统所冒的风险，本文定义并说明了基础软件的方法，特别着于基础软件与大型系统总体结构之间的关系，以及基础软件对应用系统开发生命周期的影响。

引　论

美国管理系统（AMS）解决大型应用系统技术结构的方法是根据**基础软件**，基础软件的应用戏剧性地改进了大型系统开发，运行和维护的经济性，并减少了开发这类系统所冒的风险。本文阐述基础软件概念，并从三个方面说明我们使用该方法的经验：

- 我们讨论基础软件与大型系统总体结构的关系，包括基础软件功能和组成部份，以及采用基础软件方法开发大型系统的主要受益。
- 基础软件的使用是美国管理系统指南（AMS GUIDE）：方法学，系统开发生命周期的组成部份，在系统开发生命周期每一阶段中基础软件的作用也于以说明。
- 从整个系统体系结构的观点讨论基础软和应用软件的综合。

基础软件的定义

基础软件是标准包和用户模块组成的一个综合环境，各用户模块对应用软件的开发和运行提供公共服务。这个环境对应用软件提供一个标准化的，结构化的并且简化了的外部世界视图。

采用把应用软件与计算机系统技术部件的改变隔离开以及使这些组件更易于使用的办法，基础软件增加了应用软件开发，运行和维护各个阶段的生产率。

尽管有不少反对意见，我们还是认为大多数操作系统，数据库管理系统（DBMS）远程处理鉴控程序和其它一些技术组件并不

能简化应用软件开发和运行。反而使应用软件的开发和运行大为复杂化了，基础软件方法避免了这个复杂性，并以简单有效的方式使这些技术组件对应用软件有所裨益。

软件体系结构层次

可以把大型系统分成三个主要的体系结构层次。图1示出了这些体系结构层次之间的关系。与每一层次相联系的是特殊功能和机构职责。这些层次包括：

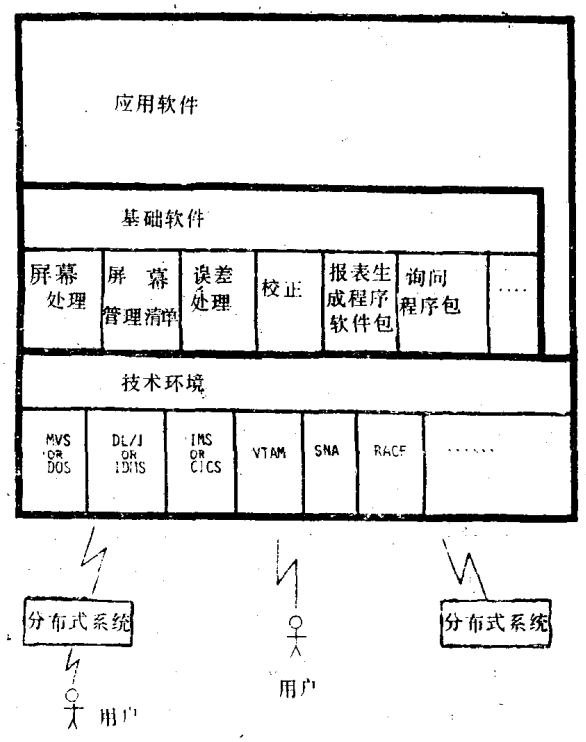

图1　系统体系结构层

1．**技术环境**：该层次包括系统设计者认为理所当然应当具备的各个技术组件，一般不能为应用目的而修改，其典型构成是

- 系统控制程序（OS／VS，DOS/VSE，MPE，VMS等）
- 存取方法及实用程序(VSAM,VTAM，IDCAMS等）
- 网络体系结构（SNA，DECNET，X.25等）
- 远程通讯监控程序（IMS DC,CICS，IDMS DC等）
- 数据库管理系统（IDMS，DL／1，Model204，IMAGE等）

对技术环境各组件的维护和支撑一般都由计算设备及其系统软件机构负责。

2．**应用软件**：本层次是应用系统的功能核心。也包括所有与现有应用（事务）问题有关的特殊直接功能：

- 应用数据编辑、检验、清除。
- 应用数据计算、分析，变换。

如果不是所有的话，也是大部分与应用数据有关的处理操作均在这一层次发生。

3．**基础软件**：本层次提供在应用软件和技术环境中每一组件所要求的详细条件之间的接口。基础软件直接使用技术环境的标准编程，通讯和控制服务，诸如数据库调用，网络信息和控制模块等以便提供像菜单处理，保密性及差错处理一类高水平公用服务。基础软件也可以包括诸如查询软件和报表生成程序等软件程序包。

基础软件功能范围

基础软件的功能范围不能严格地确定，对任何特殊的应用系统，基础软件提供的功能应根据应用软件设计特征的前后关系，技术环境的约束，和系统开发计划的机构环境来确定。

我们的经验是：把具有下列特性的功能作为基础软件的功能几乎总是更为有效的：