高性能数值软件包设计方案研究*)

谢立刚　姚继锋　陈玉荣
(中科院软件所并行计算实验室　北京100080)

摘　要　本文主要关注于高性能数值软件包的设计方案研究。在指出优秀数值软件包的几个重要特征之后，阐明了面向对象技术的引入对于设计灵活易用的数值软件包的重要意义。本文尤其着重于分析阐述基于分布式并行计算的高性能数值软件包的设计考虑，从综合的角度提出了一套框架性的设计方案。

关键词　数值软件包，面向对象，高性能，可用性，模块化，多态性，并行化

Considerations of Numerical Software Libraries Design for High Performance Computing

XIE Li-Gang　YAO Ji-Feng　CHEN Yu-Rong
(Lab. of Parallel Computing, Institute of Software, Chinese Academy of Sciences, Bejing 100080)

Abstract　This paper addresses the design of numerical software libraries for High Performance Scientific Computing. We will discuss some important features of excellent numerical libraries and explain why object-oriented techniques are introduced to support the construction of flexible and user-friendly numerical software. In particular, we give emphasis on describing our comprehensive considerations about how to achieve well-designed, object-oriented, data distrbuted parallel software.

Keywords　Numerical software, Object-oriented, High performance, Usability, Modularization, Polymorphism, Parallelization

1　引言

高性能数值计算软件包是大规模科学与工程计算中最重要和最基本组成部分之一，它的作用主要体现在以下两个方面：

1)*提高数值程序的编程效率*　与概念以及用途上的直观和准确相比，在内容上数值算法通常显得过于艰深，这导致了程序设计的复杂性。因此，代码重用对于提高编程效率有着重大的意义，事实上，基于库调用的编程已成为一种新的编程风格[1]。

2)*提高数值程序的执行性能*　对许多科学计算尤其是大规模计算问题而言，求解的正确性只是基本要求，计算的速度才是应用的关键。在数值程序的性能上，一个基本的事实是：即便对于最简单的数值运算如矩阵向量乘，普通应用者也很少有人能超越现有的标准数值库(如 BLAS、LAPACK)。因此，开发高性能数值计算软件的一个基本准则，就是尽量使用现有的经过优化的数值库[2]。

因为学科特点以及严格的性能要求，传统数值库大多采用面向过程的编程模式与函数集合的组织方式，随着应用的广泛和深入，尤其是近些年来并行计算的迅速发展，这样的组织方式无法满足复杂应用的更多要求，逐渐出现了应用危机[3]。为此，面向对象思想和技术开始被引入到数值计算领域中来，并初步解决了一些应用上的问题，目前正在广泛地研究与尝试之中[4,5]。

相对于国内高质量的面向对象数值软件包的严重短缺，国际上已有一定的研究和开发成果[6]。本文在参考和借鉴一些设计上比较先进的数值软件如 PETSc[7]、POOMA[8]、PLMP[9]、Diffpack[10]等的基础上，对基于分布式并行计算的高性能数值软件包的设计开发作了深入的研究。文中首先归纳了数值软件包的一般性设计重点，继而指出以往数值库在复杂应用过程中存在的主要问题，然后在充分考虑数值计算与面向对象软件设计思想结合的基础上，针对高性能数值计算软件包的开发提出了一套框架性的综合设计方案。文末对数值软件包的发展方向作了简要展望。

2　设计分析

2.1　主要设计重点

与众多形式各异的普通应用软件相比，绝大多数的数值软件包不管处在怎样的应用层面和抽象层次上，最后都以库软件的角色提供给用户，其终极目标是让用户实现代码重用、性能移植。影响这一目标实现的软件质量因素有很多，我们将其归纳成三个主要的方面，即功能、性能和可用性。

·*功能*　功能是整个数值软件包的核心。要设计一个数值软件包，设计者首先必须考虑的是要让软件包面向什么样的用户、针对什么样的问题、实现什么样的功能，即功能定位问题。不同的用户、不同的应用背景对软件包的期望是各不相同的，只有功能定位得当，软件包才能获得最佳效果。为此，软件包设计者必须结合实际情况，做充分的需求性分析，以决定软件包的基本构成、应用层次和技术核心。

·*性能*　对于数值计算软件尤其是面向高性能计算的数

*)本文受中科院知识创新工程信息化建设专项"超级计算环境建设与应用"项目子课题"大型数值模拟计算软件集成平台"资助(INF105-SCE-02-05)。谢立刚　硕士生，主要研究方向：高性能计算。姚继锋　博士，主要研究方向：并行算法与数值软件开发。陈玉荣　博士后，主要研究方向：大规模科学与工程计算。

值软件包，性能是绝对重要的一个指标，过低的执行性能会给软件包的应用推广带来极大障碍，这甚至通过更丰富的功能也难以弥补。为此，设计者必须对与软件包实现相关的数值计算和计算机领域有深入的了解，对其中的关键算法、关键的程序实现技术尤其要有全面和深刻的认识，以确保能采用更合理的技术方案。

·可用性　除了功能与性能，可用性也是数值软件包成功与否的重要因素。这里的可用性主要包含两方面：一是软件包的健壮性和可移植性，包括软件包对各种软硬件系统平台的兼容性和对各种执行异常的容错能力；另外就是软件包入门的难易程度和使用的可理解性、软件包适应问题的灵活性和泛化性等等。在高性能计算领域，由于高质量的并行程序开发的难度和复杂度都非常大，一个新的软件包即便可能带来合适的功能和不错的性能，如果它在应用过程中缺乏良好的可用性，为了实现代码重用需要付出太大的精力，仍然会让一般用户难以接受。

综上，功能决定了代码重用的可行性，性能决定了代码重用的必要性，可用性决定了代码重用的方便程度，三者都是数值软件包的设计重点。图1示意了普通用户对数值软件包选择的典型过程。只有具备合适的功能、出色的性能和良好的可用性，软件包才会得到用户的广泛使用。

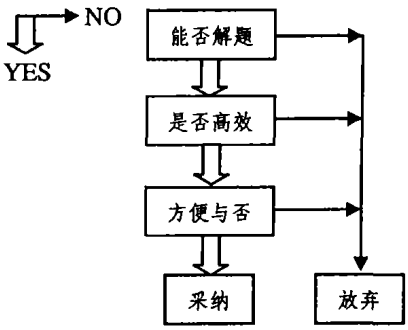

图1　用户选择流程示意

2.2　存在问题及分析

现代科学计算应用问题的特点往往表现为复杂、大型、综合，求解这样的问题大多需要基于微分方程的一整套求解技术（如图2），很多研究工作都对一种整体功能齐全、高性能、高可用的数值软件提出了强烈需求。

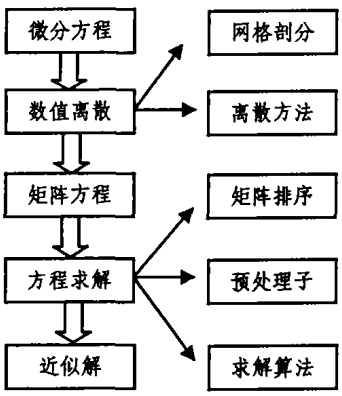

图2　微分方程数值求解主要步骤

然而以往的很多数值软件包面对复杂应用问题时的代码重用能力非常差，存在许多使用上的问题，主要表现在以下三个方面：

·通用性差　只有能让更多人使用，数值软件包才有存在的意义，为此软件包应该具有一定的通用性，但通用性差往往是许多数值软件包的最大缺陷，其主要问题在于：

1）软件实现具体化、僵化。即软件包在很多计算方法的实现过程中只简单的采取某一种具体的表现形式（如向量、矩阵的单一存储策略），而没有针对其它可能的应用进行更多的一般性考虑，从而未能给用户提供更大的使用灵活度。

2）软件结构过于简单，缺乏逻辑层次性。很多数值库都是以函数集合的形式存在的，且大多数库函数是以个体算法实现为目标而被设计成单独调用的方式，相互之间缺乏足够的有机关联度。因此，整个软件包仅仅为用户提供了核心的数值计算体，却没有为用户提供整合、重用这些计算体的高级通用平台，从而把逻辑组织的复杂性留给了用户。

软件包其功能通用性差，会给软件包的可用性带来很大的损害，用户为了实现代码重用往往需要做大量的模型调整和功能组织工作；同时，软件包应用的性能也会受到影响，那些核心计算体所具备的高性能很容易被用户不成熟的组织调用代码所带来的额外开销所掩盖。

·并行功能缺失　随着并行计算的迅速发展和复杂应用问题对大规模计算的广泛需求，高性能数值软件包应该提供可扩展并行计算的能力，而以往的大部分数值软件并不具备这样的能力，这是一个很大的不足。

·对用户不够友好　对用户的友好度是传统数值软件设计过程中很少顾及的一点，庞大的个体函数数量、冗长的函数参数列表、极少的辅助功能、不健全的文档建设等等，严重影响着用户对软件包的使用效率。

事实上，上述问题的产生与传统数值库采用面向过程的编程模式有很大关系。单一的算法被公认是很难重用的，一个通用的算法既不可能完全脱离应用的数据环境，也不应该完全限定于死板的数据假设，这就要求做到以下两点：

①为不同的应用问题定义一般性的数据结构，作为算法实现的数据基础；

②对算法的应用接口和具体执行作完全的分离，针对通用的接口封装各种具体的执行路径。

只有针对灵活宽泛、综合性强的数据结构来实现，算法才能获得良好的通用性；而算法实现的具体细节，实际上仍然离不开分类的数据假设，接口与执行的分离确保了这种底层的数据假设不会影响算法在高层使用上的通用性。

然而，以传统 FORTRAN 编程为代表的、基于简单语言变量和函数实现的面向过程编程模式很难做到以上两点，也就难以为数值软件包提供更好的通用性。与此相反，面向对象编程技术则能很好地满足上述要求，抽象数据类型、封装、继承、重载、多态等机制，不但有利于实现算法的通用平台，也有助于更方便地完成软件包的并行化，还很容易提高软件包的用户友好度。

因此，在下文的高性能数值软件包综合设计方案中，我们将充分考虑数值计算与面向对象软件设计思想的结合。

3　综合设计方案

高性能数值软件包的开发是一项庞大而复杂的工程，从设计到实现需要一个漫长的过程。目前国内数值计算界相应的开发成果还非常少，开发的经验也不多，借鉴国际上成功、流行的自由软件是必要的。在参考了多种著名的数值软件包，尤其是在使用并剖析 PETSc（Portable，Extensible Toolkit for Scientific Computation）[11]的基础上，结合并行数值程序

设计的应用经验，我们针对高性能数值计算软件包的开发提出了一套框架性的综合设计方案。

3.1 实现软件包的模块化和功能集成

有两种极端的软件包存在形式：一是“散装”形式，软件包提供形形色色的充足的库函数，以备用户自由调用，而很少为用户组织这些库函数；二是“黑匣子”形式，软件包直接针对某个特殊的使用目的对整个实现过程做了全封闭的封装，只能按“输入参数——输出结果”的方式使用。前一种方式可以实现最充分的代码重用，但是把软件包使用的复杂性完全留给了用户，后一种方式实现了最高的功能集成，但丧失了调用的灵活性，这对其内部的很多函数而言是个很大的浪费。除了极少数特殊的用途（数值核心库、终端产品等），这两种方式都是不足取的。

模块化方式介于上述两种形式之间，在模块内部实现封装，在模块之间实行有层次的“散装”。封装的意义在于提供功能集成接口、屏蔽具体实现细节，让用户更多地以功能调用的直观形式获得代码的重用，而无须把精力耗费于底层函数的重复组织上。“散装”的意义则在于实现功能的层次性，让模块在理论逻辑的指导下和功能接口的规范下，具有单独使用和组合使用的充分自由，从而确保了软件包处理不同应用问题的灵活性。模块化方法是对两种极端形式的权衡折中，这种折中是完全符合实际的，因为绝大多数应用需求是既有共性又有个性的，良好的工具应该对共性做统一、对个性保持灵活。

为了更好地实现上述意义下的折中，软件包必须进行合理的模块设计，包括整体上的体系结构设计、外在的接口设计和内在的功能及算法实现设计等等，这通常是一个具体而复杂的问题，必须基于对特定应用背景的深刻认识和对理论体系的全面把握。而从一般性的角度来看，良好的数值软件包模块除了数据封装、细节隐藏、实现功能、提代接口外，还应该具备两个重要的特点：

·单元性　这里的单元性有两层意思：一是功能独立性，即模块自身的功能在实现上是独立的，这确保了模块调用的安全性，任何一个满足模块接口规范的调用都是合法的，而不管其它调用环境；二是功能单位性，即模块的功能既要有一定的集成度，也要尽量保持在一个抽象层面上，避免过于单薄或者跨度过大甚至混乱交叉的功能组织，这确保了模块组织调用的直观性和灵活性，一个重要的例子是高层的算法必须从底层的数据结构中分离出来。

模块的功能独立性是必须保证的，这要求模块在设计和实现上必须遵循“强内聚，弱耦合”的设计原则。模块的功能单位性则是相对的，它体现了软件包设计者对特定应用背景数值程序设计的把握，需要从体系结构设计的角度对软件包进行合理的层次设计。

一个非常自然的想法就是按数学理论的概念层次封装各抽象数据类型，进行数学概念层次模块化。这种数学概念层次包括数据结构层、算法层、应用层等上下层次，对同一层次还可以继续分解为上下层或者相邻层次，每一个层次都对应一个抽象的数学概念范畴，包含一些同类的数学对象，从而适合封装成一个模块。比如，对于线性代数方程的迭代求解，有两个基本的概念层次即预处理方法和迭代方法（图3），每一个层次都包含非常多的内容，但由于其在概念上同属于一个层面，可以分别封装成一个模块。

方程求解	预处理	Jacobi	ILU	ICC	…
	迭代法	CG	GMRES	CGS	…

图3　线性方程求解的基本概念层次

·多态性　数值程序设计的多态性需求来源于数学系统本身蕴含的广泛的多态性，即一个抽象的概念或过程常常对应于多个不同的具体形态。数值软件包的各个模块应该充分模拟各种数学概念之间所具有的多态特质，尽可能为用户保留“抽象性”，让数学概念成为维系用户与模块的天然纽带，使用户站在更高的层次上以数学直观的形式使用模块。

软件包多态性将给数值程序设计带来的主要好处在于对同类功能的抽象集成和对不同操作的自动区分。把一系列完成同一目标的具体行为集成到一个抽象的行为，让抽象行为的接口既不依赖于具体算法的选择，也不依赖于数据存储策略的选择，这种接口与执行的分离可以实现用统一的程序结构执行不同的数据路径。比如矩阵向量乘法，矩阵可以是连续存储的稠密矩阵，也可以是压缩存储的稀疏矩阵，它们在具体实现上是不同的，但可以提供一个统一的矩阵向量乘法操作接口。又如对于图3所示的线性方程求解，可以提供一个通用的求解器接口，具体执行过程中采用哪种预处理和迭代方法，由用户传入的参数类型决定。

这种抽象集成对数值程序有非常重要的价值，因为对一个具体的问题求解，在很多情况下，没有一种执行路径在每次执行前就被认为一定是最佳的，用户通常需要对不同的选择作多次的尝试和比较，通用的程序模块无疑大大提高了编程的产出投入比。

面向对象程序设计技术为模块化设计的上述所有思想都提供了技术支持。抽象数据类型可以很好地模仿数学对象的内涵和行为，封装、组合和继承机制能很好地完成层次清晰、搭配协调、接口简明、功能集成的模块，模板类、重载、虚函数等技术则能有效地实现软件包内在的多态。

3.2 提供并行的功能隐藏并行的细节

面向高性能计算的数值软件包的一个基本要求是软件包应该提供并行计算的能力，使得软件包能够在多处理机上运行以适应大规模的科学与工程计算问题。

大规模计算问题对软件包的可扩展性提出了很高的要求，即随着参与计算的处理机数量的增多，数值程序应该获得稳定增长的加速比。相对来说，消息传递编程模式比统一地址空间编程模式具有更好的通用性和可扩展性，尤其是网络互联技术的高速发展、PC—Cluster 硬件平台的广泛流行，使得采用消息传递标准接口 MPI 进行分布式并行编程是目前数值软件包并行化的首选。

并行算法的设计是一个非常复杂的过程，消息传递并行编程也不是一件简单的事情，一个并行程序的开发过程既需要充分挖掘问题的并行度以进行计算任务的合理分配，还需要处理很多琐碎的数据通信以保持分布计算的必要关联。因此，数值软件包一方面要对内部模块的功能实现并行化，一方面必须替用户隐藏并行实现的技术细节，这种细节隐藏对用户而言无疑是极具意义的。

分布式并行计算的实质在于数据的分布存储以及对分布数据的协同处理，只有明确了数据的分布状态，才能正确执行其上的操作。因此，软件包的并行化必须从底层的数据结构模块着手，一个自然的处理办法即通过数据结构并行化的思想，定义能够分布存储的数据类型，如分布存储的向量、矩阵（如图4）、网格等，并以此为基础，完成整个软件包所有模块的并行化。鉴于底层数据对象是整个数值软件包的基本操作元素，

这种并行实现途径的可行性是有保证的。

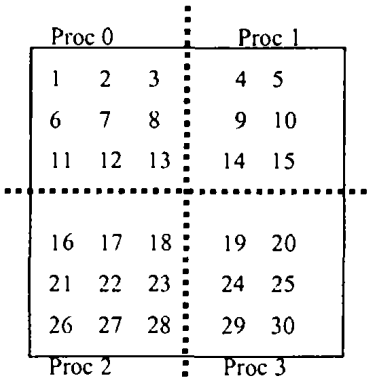

图4 分布存储的矩阵示例

同时，为了更进一步隐藏通信细节，还可以为分布存储的数据对象提供一套全局的索引，使用户对它的存取访问与串行模式保持一致。事实上，这种可分布存储的数据类型的实现，仅仅需要对原串行数据类型做数据成分的扩充以提供必要的标志信息，即可获得两种类型的统一，而无需另起不同的定义。于是模块的并行化并不需要改变调用接口的形式，即无论并行还是串行，接口界面是完全一致的。此时，多态机制获得了更大的意义，区分执行是否并行的信息将由调用功能接口的具体数据对象所提供，这是由对象被创建时是否分布存储决定的。

由此，软件包实现了并行机制的细节封装，提供了自动并行的功能，在用户眼里，大部分数据对象和算法的并行操作都具备整体性和全局性，用户只需从宏观上了解并行执行的事实，而无须关注并行实现的技术细节。

3.3 从多种途径追求软件包的高性能

如前所述，追求更高的性能，是所有数值软件包的重要目标之一，这个目标贯穿于软件包开发的始终，为此，软件包设计者必须从算法和程序实现两方面做努力。

·算法。算法对于应用问题的重要意义是显而易见的，如果说计算机硬件条件在应用中起着重要的作用，则算法的选取才往往是应用的关键。与发展相对平稳的计算机技术相比，算法革新常常能带来数量级上的性能改进。正因为对同样的求解目标，不同算法的求解效果千差万别，算法才成为了数值计算的核心内容。数值软件包所做的一切工作，从某种意义上来说是在提供一个算法重用的高级通用平台。

因此，一个数值软件包的开发，首先要在算法研究上有所作为，在国际上一般性的自由软件包比较丰富的情况下，简单的重复劳动是不足取的。这就要求开发者既要对软件包所关注数值领域的前沿动态有充分的了解，又要力争在某一方面有所突破。尤其是面向工程应用的实际问题，理论界的一般性算法还有改进的余地，这取决于对问题特殊性的把握，往往体现在若干个关键参数的选择上。

·程序实现。程序实现同样是一个必须关注的重点，因为一些编程技术细节对软件包的性能也会产生重大的影响。为提高程序的执行效率，主要应从如下三个层次着手：

1）核心计算代码的优化 与普通程序相比，数值程序最大的特点是其大部分执行时间都用在了浮点运算上，而且产生这些浮点运算的代码往往是只占程序总量很小部分的内层循环体。因此，越是内层的核心计算代码，越要注意细节的代码调度。一个代码优化的捷径是对基本的计算任务尽量调用成熟的数值核心库（如 BLAS、LAPACK），让数值核心库成为软件包的基本组成部分。

2）核心通信代码的优化 在并行计算过程中，数据通信所带来的性能开销是并行执行所付出的代价，但这种代价有时显得过于严重，而良好的程序实现方案能降低无谓的性能开销，比如对特定的消息传递采用更佳的通信模式，通过不相关代码的前置获得通信与计算的重叠，通过局部的小规模计算代替远程数据的发送接收等等。

3）防止过分面向对象设计带来的性能损失 如何避免由面向对象程序设计方法带来的对数值软件包性能的损害，是最近几年科学计算的一个新的研究课题[12,13]，在软件包的开发过程中需要特别的关注。当然，这跟不同编程语言的特点及其编译器的发展程序有关。一般性的研究结果表明，适量的使用模板类、重载等编译时多态对性能没有影响，而只要把以虚函数为代表的运行时多态尽量用在构建高层结构上，禁止对循环体内的核心计算代码使用，对软件包整体性能产生的影响也很小。当然，还要注意尽量避免自定义数据类型的构造函数所产生的不必要的临时变量。

3.4 以用户为中心提高软件包的可用性

如果说面向对象设计思想和开发技术已经从多个方面对数值软件包的易用性产生了质的改观，则一些非面向对象因素对软件包的可用性也有重要的影响。为提高数值软件包的可用性，必须要以用户为中心，把更多的努力投入到对用户需求的支持上，这主要有以下几个方面：

·可移植性。可移植性是数值软件包的基本要求之一，与普通应用软件具有相对稳定的使用平台不同，高性能数值软件包必须适应用户各不相同的计算环境，这就需要软件包对支撑系统的特殊要求尽量地少，因此，软件包在开发过程中一定要注意使用标准化的语言、库函数和消息传递接口。

另外，对于一些不可避免的环境特殊性所带来的软件包使用问题，设计者有必要对特定的系统分别进行专门的、有针对性的底层配置，并提供一些基准测试程序。

·接口对应用的合理性。接口无疑是面向对象软件包的设计重点，一个著名的开发原则是“面向功能接口，而不是面向功能实现”，因此接口的设计显得尤为重要，必须充分地考虑应用的需求。为了防止软件包主观的设计目的与用户的实际需求不相符，设计者应该与用户建立广泛的联系、接收使用的反馈信息、实现与用户的双向交流，以提高接口对应用的合理性。

·服务综合度。用户在使用软件包进行数值计算的过程中，往往会有一些其它的功能需求，比如有效的程序报错机制、高级的性能调试方法、必要的可视化功能，有的还希望以命令行输入参数的形式方便地指挥程序的多态执行以及在计算过程中获得浮点性能统计，等等。有选择有特点地为用户考虑这些功能，将给软件包带来更大的使用价值。

一个容易忽视但却尤其重要的问题是软件包的文档建设，包括用户手册、使用示例、入门教程、技术支持等等一系列形式的文案，这些说明性的文字往往是用户是否选择软件包的重要参考因素。

·开放性。这里所说的开放性是对用户来说的，主要有与常用软件包（如 MATLAB）的联用能力和接受用户自定义函数模块的能力等。例如，很多时候，某种算法的缺失会给软件包模块的使用带来障碍，接入用户自定义算法是一种必然的需求。为方便同类算法的接入，软件包在自身功能实现的同时为用户预留规范的自定义函数提交接口，使得用户不必去读复杂的软件包源代码并试图做危险的修改，即可正确方便地实现自定义函数的接入，这在实际应用过程中是很有意义的。

数值软件包的这种开放性保证了它强大的适应性和持久的生命力。

总结与展望 上述思考仅给出了对高性能数值软件包设计方案的框架性意见，尚未涉及具体的软件包开发实践。事实上，优秀的高性能数值软件包的开发极具挑战性，一大困难之处在于它对开发者的数学修养和计算机素质两方面的要求都非常高，理想情况下需要计算方法领域专家、计算机系统专家、软件设计师、有经验的程序员、测试员、专业用户的集合体才能胜任，因此团队的合作是不可缺少的。

最近几年，为了获得数值软件包更高的可复用性，国际上关于不同数值软件包的一体化工作正在广泛地进行，其中一个重要的进展是面向构件的设计思想[14]，框架、构件、设计模式这些软件工程领域时髦的概念正在深刻地影响着数值软件包的设计理念。简单地说，构件是一种比模块更具重用性的功能封装体，数值构件的设计目标是实现一种跨语言、跨软件包、跨平台的软件单元，接口是其唯一的使用途径。构件技术的标准化以及高可复用性的数值构件库的广泛开发，必将给数值程序设计模式带来质的飞跃，未来的数值程序将以在一组构件中挑选合适的构件进行动态组装的"搭积木"方式进行。

参 考 文 献

1 Bjarne Stroustrup. C++ Programming Styles and Libraries. Copyright Informlt. Jan. 2002

2 Grant M C. Numerical linear algebra software. Oct. 21, 2003 Web site: www. stanford. edu/class/ee392o/nlas. pdf

3 Gropp W D. Why we couldn't use numerical libraries for PETSc. In: Proc. of the IFIP TC2/WG2. 5 Working Conf. on the Quality of Numerical Software, Assessment and Enhancement, 1997. 249～254

4 Boisvert R F, Moreira J E, Philippsen M, Pozo R. Java and numerical computing. IEEE Computing in Science and Engineering, 2001, 3(2): 18～24

5 Kees C E, Miller C T. C++ implementations of numerical methods for solving differential-algebraic equations: design and optimization considerations. ACM Transactions on Mathematical Software, 1999, 25(4): 377～403

6 Object-Oriented Numerics Page: http://www. oonumerics. org/

7 PETSc home page: http://www-unix. mcs. anl. gov/petsc/petsc-2/

8 POOMA home page: http://www. codesourcery. com/pooma/pooma

9 PLMP home page: http://www. hpcl. cs. msstate. edu/pmlp/

10 Cai X, Langtangen H P. Developing Parallel Object Oriented Simulation Codes in Diffpack. WCCM V, Vienna. Austria, 2002

11 Balay S, Buschelman K, Gropp W, Kaushik D, Mclnnes L C, Smith B, Zhang Hong. PETSc Users Manual, Argonne National Laboratory, 2002: [Technical Report ANL-95/11]. Revision 2. 1. 3

12 Bassetti F, Davis K, Quinlan D. Toward FORTRANN7 Performance from Object-oriented C++ Scientific Frameworks. In: Proc. of the High Performance Computing Conf. 1998

13 Budimli'c Z, Kennedy K. The cost of being object oriented: A preliminary study. Scientific Programming, 1999, 7(2): 87～96

14 CCA Forum home page: http://www. cca-forum. org/

15 http://www-unix. mcs. anl. gov/～gropp/

16 MPI Forum home page: http://www. mpi-forum. org/

（上接第139页）

其余属性和过程定义相同。活动接口列表包含了所有在活动模板中使用到的活动接口。过程模板本身也可以和过程一样扩展其它的过程模板。

增加活动中的活动接口属性(Activity Interface Id)。如果活动接口属性值不为空，表明该活动实现了此属性值指定的过程模板中的活动接口；反之，则表明该活动不实现任何活动接口。当实现某一活动接口时，此活动自动具有这一活动接口中已经定义的属性值，如开始状态(Start Mode)、结束状态(Finish Mode)、限期定义(Deadline)和转移信息(Transition Restrictions)等等。在活动中也可以重新指定这些已经在活动接口中定义了的属性值。

增加活动接口元素(Activity Interface)。活动接口在活动的基础上，删除了其中的活动接口属性(Activity Interface Id)和活动实现部分描述(Implementation)，其余属性和活动相同。在活动接口中定义的属性值将被传递到实现其的活动中。

扩展可以引用活动的其它元素，使可以引用活动的地方也同样可以引用活动接口。如对于转移信息(Transition Information)，扩展其中的起始活动属性(From)和终止活动属性(To)可以被指定为活动接口或者活动。

结论 本文在WfMC定义的工作流过程定义模型和过程定义语言的基础上，通过引入模板机制，提出了一种基于模板的工作流过程定义方法及其相应的定义语言。使用该方法描述的工作流过程具有较强的可复用性和扩展性，同时也具有一定的动态变更能力。目前，这一工作流过程的定义方法已经被应用于一项正在实施的软件研发项目中。该软件项目，即SPIF(Software Process Improvement Framework)系统，是一个基于Web B/S结构的面向CMMI的企业级软件过程改进支持系统，其目标是为了辅助软件企业在CMMI规范的框架下实施软件过程的管理和改进，目前正在进行第二版本的设计和开发工作。此项目已经开发出一个简单工作流引擎，支持采用该基于模板的定义方法实现了软件过程的定义，在实际应用中取得了很好的效果。

参 考 文 献

1 Paulk, Mark C, Curtis, Bill, Chrissis, Chrisis M, Weber, Charles. Capability Maturity Model for Software Version 1. 1. Software Engineering Institute. CMU/SEI-93-TR-24. 1993

2 CMMI Product Team. CMMI-SE/SW/IPPD/SS Version 1. 1. Staged Representation. Software Engineering Institute. CMU/SEI-2002-TR-011. 2002

3 Workflow Management Coalition. The Workflow Reference Model. Document Number TC00-10003. 1994

4 Workflow Management Coalition. Interface 1: Process Definition Interchange Process Model. Document Number WfMC TC-1016-P. 1999

5 Workflow Management Coalition. Workflow Process Definition Interface -- XML Process Definition Language. Document Number WFMC-TC-1025. 2002

6 范玉顺，吴澄. 一种提高系统柔性的工作流建模方法研究. 软件学报，2002，13(4)：1～2