层次化软件构造

龚洪泉　邱晓娜　薛云皎　钱乐秋
(复旦大学计算机与信息技术系软件工程实验室,上海 200433)
E-mail:springgh@263.net

摘　要　该文针对面向对象和可视化开发环境中出现的常见问题,结合软件系统开发的实践经验,参照计算机网络中的层次参考模型,提出了层次化软件构造的思想。同时结合实现软件复用的构件技术,将软件系统的组成构件划分为物理操作层构件、公共服务层构件、特定领域层构件和用户界面层构件4个层次。文章最后结合软件开发项目的具体应用,说明了层次化软件构造的有效性和实用性。

关键词　面向对象技术　可视化开发环境　计算机网络层次参考模型　层次化软件构造　软件复用　构件技术

文章编号 1002-8331-(2003)05-0135-04　**文献标识码** A　**中图分类号** TP31

Layered Software Construction

Gong Hongquan　Qiu Xiaona　Xue Yunjiao　Qian Leqiu
(Department of Computing and Information Technology,Fudan University,Shanghai 200433)

Abstract: In this paper,some problems related to the object-oriented technology and the visualized development environment are addressed.Based on the practice experiences of software system development and the computer networks layered reference model,the method of layered software construction is presented.Using the component technique,components of software system are grouped into physical operation component,common utility component,specific domain component and user interface component.At the end,according to the successes of accomplished software system,the usefulness and practicability of layered software construction method has been proved.

Keywords: Object-Oriented Technology,Visualized Development Environment,Computer Networks Layered Reference Model,Layered Software Construction,Software Reuse,Component Technique

1　引言

随着面向对象技术日趋成熟和可视化开发环境的大量使用,软件质量在一定程度上有所提高。软件可靠性、可扩展性、易维护性等都得到了加强,软件的开发和维护负担有所减轻。然而在大多数的软件设计开发过程中,软件设计开发人员往往无意识地忽视面向对象技术为人们带来的好处。他们仍然在采用传统的系统分析、设计和开发方法。这并不意味着传统方法没有长处,而是应当在传统方法的基础上尽量利用面向对象技术为大家带来的各种好处。这也并不是说软件设计开发人员不想利用这些好处,而是他们可能因为可用的选择方案而受挫,转而回到他们以前所使用的非面向对象的设计技术[9]。面向对象技术并不是灵丹妙药,面向对象技术只是给大家提供了一种好的软件开发机制,如何去挖掘面向对象技术的潜能,如何在软件设计开发过程中有意识地利用面向对象技术,则需要形成一些新的软件构造思想。

2　可视化开发环境中的常见问题

随着软件开发平台提供的各种文档自动管理功能,用户界面生成功能以及构件拼装功能极大地减轻了软件开发人员的开发负担,有力地推动了可视化开发环境的使用。有了可视化开发环境,在开发一个软件系统时,开发人员不必再为系统常用界面的生成花费精力,他们在系统输入、输出界面如何布局的问题上只须稍作考虑,而把大量的时间投入到系统逻辑功能的实现上。这样不仅节省人力物力,缩短软件开发周期,也使软件系统的质量有所提高,因为开发人员可以花更多的时间来考虑如何提高系统的质量。

然而,正是可视化开发环境的使用,常常看到很多开发人员在实现系统功能时忘记软件模块化的原则。他们通过系统设计文档得知在比如系统时钟或用户输入等消息之后软件系统应当做的一系列内部操作,于是他们便把完成这一系列操作的所有代码写到一个事件的处理过程中。这种开发方法表面上看起来是理所当然的,因为它把相关的东西放在一起,看起来还有点'高内聚'的风格,而且当别人来读一段代码时也很容易知道某个事件后系统作了哪些操作。

可是这种方法没有彻底地贯彻软件工程中模块化的思想,大量的代码揉杂在一起不利于系统的维护。系统的流程控制被淹没在大块大块的代码行中,使得系统维护人员甚至使开发人员自己也看不清系统的体系结构,无形中增加了系统的维护负担。更为严重的是这种方法不利于软件系统的扩展,由于系统条理不清晰,当系统要添加或删除功能时,开发人员根本无从

作者简介:龚洪泉,硕士研究生,研究方向:软件工程,软件复用,构件库管理系统,基于构件、构架的软件开发方法,系统开发模式。邱晓娜,微软亚洲技术支持中心,操作系统技术支持工程师。薛云皎,硕士研究生,研究方向:软件工程,构件库管理系统,基于构件、构架的软件开发方法。钱乐秋,教授、博士生导师,软件工程实验室主任,研究方向:软件工程,软件复用,构件库管理系统,基于构件、构架的软件开发方法,基于代理的软件开发,系统开发模式。

知晓该到哪个地方去进行改动。对这种情况,也许添加功能还好一些,因为功能是累加的,添加新的东西不一定会影响原有的功能特性。但当系统的某个工作流发生变化时,对系统的改动可能是灾难性的,因为这样有可能影响系统原本十分脆弱的根基。第三个方面,这种方法不利于软件复用,这与软件界的软件复用潮流是相悖的。也就是说在一个软件系统中或多或少存在一些可以多次使用的底层或中间层操作。在上面提到的这种有问题的方法中,当遇到一个有类似功能的新操作时,开发人员往往将原来的代码拷贝过来,然后将新操作需要的特殊功能添加进去。这种方法看起来节省了不少的时间,在很短的时间里就实现了系统新的功能,而且也有点象软件复用中提到的代码级复用。但大家知道,这是一种十分危险的行为,它增加了大量的系统维护负担,甚至有时会给系统带来灾难性的后果。当在维护过程中发现原来的实现中有问题时,不仅要修改最初的实现模块,而且要修改系统中该模块的所有拷贝,这是一项十分繁重的维护任务。可怕的是由于不知道系统中这个功能模块到底有多少份拷贝,如果有一些地方没有修改到,那么这种后果有可能是灾难性的。

上面列举的只是一些笔者在可视化开发环境中遇到的常见问题。问题的根源不在于可视化开发环境,而在于一些软件开发人员缺乏系统的软件开发思想的指导,从而无意识地陷入了软件开发的误区。为了避免软件开发人员陷入这种尴尬境地,这里提出了层次化软件构造的思想。

3 层次化软件构造

按照一般的开发思路,软件开发人员无法完全避免上述问题,因为他们完全是根据系统的设计来实现系统的。为了使开发人员在开发软件时对系统体系结构有一个清晰的把握,不至于将一大堆工作混杂在一起而造成系统的维护困难,借用计算机网络中的层次参考模型将软件系统进行一个层次化的分类[4]。软件系统的最终目标是完成目标系统需求所要求完成的任务。系统的各个组成部分在系统中扮演不同的角色,完成不同的任务,它们组合在一起共同完成软件系统的所有任务。其中有些部分实现比较底层的操作,比如控制一个物理硬件的运转或与数据库打交道;有些部分实现某一类数据结构的基本操作,比如栈、队列的操作等 还有些部分是专门与系统的最终用户进行交互,读取用户输入信息,将结果反馈给用户等。这些不同部分之间有可能是纯粹的聚集关系,它们在一起只是因为系统需要它们各自的功能来完成不同的任务。另外一种关系就可能是层次关系,实现底层操作的部分为实现较高层操作的部分提供服务。有了底层的基本服务,高层操作就只需要关心本层必须完成的任务。对于底层的操作,它只需要明确定义可以提供给高层的服务接口即可, 其内部可以采用不同的实现方法,层与层之间是完全透明的, 它们只通过层间的服务接口进行交互。

在对一个新开发的软件系统进行分析后首先对它的所有功能进行层次化划分,然后对每一个层次的功能进行模块化分解。一个层次中可能有多个模块具有相似的功能,对这些模块进行更深层次的划分,将相同部分提取出来作为低一层次来对待,而上层的不同部分就划分到不同模块中。这样的层次划分和模块化分解一直进行下去,直到系统所有功能都有一个明确的模块归属为止。这种软件系统的分解采用先水平后垂直的划分方法 如图 1 所示),层与层之间的界限是明确的,但层中的模块内部可能要进行进一步的划分,这是一个反复迭代的递归分解过程。至于为什么不采用先垂直后水平的划分方法是因为先垂直后水平的方法与传统的 "自上而下,逐步求精 "方法类似,这种方法不易识别系统底层的公共操作,减少了底层模块的复用机会,容易造成不必要的重复开发 如图 2 所示)。

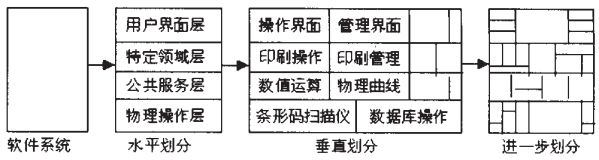

图 1 先水平后垂直的划分方法

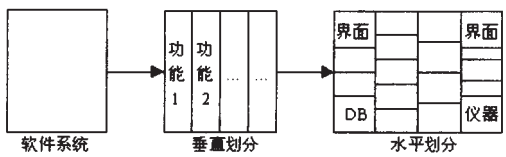

图 2 先垂直后水平的划分方法

在对软件系统进行先水平后垂直的分解过程中,可能会发现同一层不同模块的内部模块中有功能相似的子模块,对于这种情况需要将它们的公共部分进行提取,然后对初始的层次划分进行调整,将公共部分作为它们的下一个层次,当前层次只需要去访问公共部分提供的服务 如图 3 所示)。

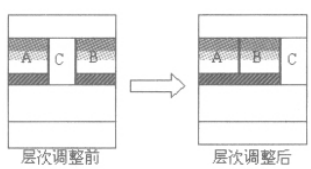

图 3 层次调整

完成软件系统的分解后,首先按照各模块对上层提供的服务来定义它与上层模块之间的接口。定义好层与层之间的服务接口后,就可以对各个模块进行独立的设计了。有了这种先进行层次化分解再进行独立设计的方法,软件开发人员就可完全按照各模块的设计进行独立的开发,这样就不会再出现将多个层次的东西揉合在一起的情况。

4 基于构件的层次化软件构造

软件工程的目标是生产高质量的软件产品,但如何充分重复利用过去成功软件中好的设计和开发思想从而确保新开发软件产品的质量,基于构件的开发提供了一种可行的方法。构件是 "一个内聚度高的软件包,可作为一个单元独立开发和发布,并且能方便地与其它构件组成更大的构件或系统 "[1]。这里不深入探讨构件的内部机理,只是将当前流行的构件技术作为一种工具来使用,以便能将过去成功软件中好的设计和开发思想带到新的软件产品开发过程中,以使层次化构造软件的思想在实践中更容易实施。

与工业生产中一个大的产品由一系列零部件组装而成相类似,基于构件的软件开发就是使用一个个定义良好的实现了的且经过严格测试的构件来搭建新的软件系统。待开发系统需要的构件可能在以前已经开发过,所以不必让所有的构件都从

头来开发。对于过去成功软件产品中的构件,可以直接拿过来应用到新的软件产品中,这不仅节省了设计开发时间,而且由于这些构件都经过严格的测试,新的软件产品质量就能得到很好的保证。这就是软件复用思想的体现,构件技术就是直接为了软件复用而在软件工程领域出现的新技术。开发构件的目的是为了复用,而层次化软件构造方法则是为了充分利用构件技术而提出的。

当对一个软件系统进行分解并完成了各个模块的设计后,将着手把各个模块设计成构件,以利于以后复用。这些构件中处于较高层次的构件可以认为是特定领域的构件,因为它们与当前开发的软件系统关系相对较紧密。而处于较低层次的构件只是与物理硬件或数据库打交道,或是提供一些通用算法的实现,它们可被认为是公共服务构件,因为在不同软件系统中都可能使用它们。

为了明确层次化软件构造过程中不同构件所处的层次,依照软件系统的层次分解思路将构件归属于如下几个层次:

(1)物理操作层构件:这类构件与物理硬件或数据库打交道,其主要功能是完成物理硬件的控制或数据库操作。划分这一层构件的目的是用这一层的构件来屏蔽底层硬件操作或数据库操作的复杂性和多样性,为上层应用提供统一的透明的硬件或数据库操作服务接口。

(2)公共服务层构件:这类构件专注于提供常见问题的高效解决算法,同时提供与操作系统相关的公共服务,比如文件读写、目录管理、网络通信等。它还对一些通用数据结构提供常见的操作支持。这类构件支持水平复用,也就是说各种软件系统都可能使用它们提供的服务,从而避免重复劳动,节省软件开发时间;避免重复开发过程中引入错误,提高软件产品质量。

(3)特定领域层构件:这类构件与特定应用领域有关,它们提供该应用领域中常用的数据结构及其操作算法,为应用领域搭建基础设施。这类构件支持垂直复用,也就是说在同一领域中开发新的产品时可以大量复用这些构件。划分这一层构件的目的是为了避免开发类似产品时的重复劳动,是为了开发系列产品时重复利用原有的成果,从而缩短新产品的上市时间,赢得更多的客户支持,占有更多的市场份额。

(4)用户界面层构件:这类构件与软件系统的最终用户打交道。提供用户界面操作服务,可以说它们也属于公共服务这一类。但由于这些服务有许多自身的特点,所以将它作为单独的用户界面层构件来讨论。当前可视化开发环境中已经提供了大量的界面构件,为软件系统的界面开发提供了有力支持。可以对这些界面构件进行组合、改进,开发不仅使最终用户感觉友好,而且使开发人员更易于使用的大粒度用户界面构件。

这4个构件层次是一个大粒度的划分,它们之间不具有严格意义上的上下层关系,在各个层次构件的内部层次划分中才能看到明显的上下层依赖关系。一个软件系统的层次化构造在粗粒度角度看是这4个层次构件的聚集,当从细粒度角度,即从内部去观察软件系统时,就可以看到内部明显的层次关系。

之所以提出基于构件的层次化软件构造是因为构件技术在很大程度上能使软件复用思想得到具体应用。采用基于构件的技术,有利于系统可扩展性能的提高,使系统具有更大的灵活性。当完成一个软件系统所有构件的开发后,系统集成只须按照系统需求将构件组合起来。要添加或删除系统功能只须将相关构件加入或从系统中去除就行了。这大大减轻了软件系统的维护负担,并在很大程度上保证了软件系统的可伸缩性,为开发类似产品或系列产品提供了有力的支持。

下面对上面提到的4个层次的构件作进一步的描述:

5 物理操作层构件

这类构件主要用来为上层提供控制物理硬件或数据库的统一的、透明的操作服务,屏蔽底层操作的复杂性和多样性。对于操作物理硬件的构件,要定义硬件控制信息的数据结构,控制信息的编码规则,硬件反馈数据的编码规则等。同时要定义这些构件为上层提供的服务接口。这些接口必须明确定义,并且简单清晰,这样只要保持接口不变并提供原有的实现相同的服务,就可以用一个完全不同的实现来替换原有的实现。这些要求是与对构件接口定义的要求一致的,因为最终的目的是只要它们提供了完全相同的服务就可以自由地替换构件。对硬件的一些常用操作可以用统一的方式加以定义,比如打开或关闭硬件、发送控制信息、读取反馈数据、捕获硬件中断请求等。对各种硬件用统一的方式为上层提供服务,简化了硬件操作的的复杂性,减轻软件开发人员的开发负担。

对于特定领域使用的硬件,比如高炉控制中的温度计,用于产品记录的条形码扫描仪,声控设备的声音转化装置,手写输入的手写板等,需要为它们设计特殊的控制逻辑模块。将这些模块开发成构件,在设备升级或开发系列产品时,只须将对应的构件集成到硬件的控制软件中。在硬件设计不变动的情况下,只要更新相应的控制构件,就可实现新的功能或改进硬件原有的性能。

对于数据库操作构件,承袭ODBC的思想,为数据库选择、插入、删除、更改等常用操作提供统一的操作接口。不管后台数据库类型、大小如何变化,前台的应用软件不需要任何变动仍然可以完成系统要求的任务。对这类构件可集中在开发高效的数据库引擎方面。

6 公共服务层构件

公共服务层构件主要应用领域有操作系统的底层操作支持、网络通信、数据传输、科学计算等。在操作系统底层操作中主要有文件读取、目录管理。网络通信、数据传输方面主要是提供实现了某种通信协议的操作简便的构件,比如HTTP、FTP、SMTP等。

在科学计算领域,比如三角函数中正弦、余弦、正切、余切等求值的高效算法;高精度数值运算的处理;对数求值;微分、积分运算;常见数值分布的概率求解等等都可根据各领域专家提出的高效算法来设计成相应的构件。这样就不需要每次遇到类似计算时都先去查阅资料,然后确定算法,最后来编码实现,既可以节省新的设计开发时间,也能确保算法的正确性。

在科学计算领域中的另一个重要方面是各类函数曲线的绘制。比如常用的正弦、余弦、圆、矩形、抛物线和各种物理实验曲线;化学中的分子、原子结构图;生物学中细胞变化轨迹等都需要绘制出来以给观察者一个感性的认识。而这中间许多结构的绘制过程是十分复杂的,把这些绘制算法开发成特定领域的绘图构件,可以为这些领域的专家解决大量棘手的问题,使他们只须关心自己研究对象的变化机理,而不必为他们不熟悉的计算机问题花费不必要的精力。

另一类公共服务构件主要是提供常见数据结构的操作支持,比如栈的进栈、出栈、获取栈顶元素、判断是否栈空等操作;还有一组元素的排序、在一组元素中查找指定元素的位置、在

一组元素中插入新的元素或删除一个指定的元素等操作。无论元素的类型是整数还是组合结构，无论元素的存放方式是数组、链表还是树，这些构件都可以提供统一的操作服务接口，并根据不同的数据结构动态选择高效的实现算法。有了这些构件，就不需要在开发程序时无数次地写类似如下所示的查找语句了：

```
FOR index :=1 TO MaxIndex DO
BEGIN
 IF (.=...[index]) THEN
  ...
 ELSE
  ...
END
```

公共服务层构件在软件开发过程中使用频率很高，这类构件的开发需要作仔细深入的分析，并结合领域专家的研究成果来进行。

7 特定领域层构件

这类构件与实际的应用领域密切相关。首先需要领域专家对该领域作广泛深入的调查研究，进行领域分析，搜集各种资料，最后为各个构件界定范围。比如在财务软件领域，分析专家必须对各种凭证、账单和报表的来龙去脉有一个清晰的把握，并且还要求对国家的财务法规制度有透彻的理解，这些构件的工作流程必须在法规制度允许的范围内开展。在印刷出版行业，各类出版物有不同的排版规则，除了遵循通用的排版准则外，这类构件应提供尽可能多的排版选择方式。对于不同的排版方式，可以开发一系列构件来供印刷出版商选择。各行业的统计信息，报表、明细账打印等都有各自的格式，对这些常见的用户需求进行分析后可以开发通用报表打印构件，然后设置一些变化点来适应不同的应用环境。在有明显前后流程制约的行业，比如自动柜员机上的取款操作，超市购物过程等，可以使用系统用例（use case）来指导构件的开发[3]。

特定领域层构件的开发有利于为开发类似产品和系列产品节省领域分析、设计和开发时间，并节省测试所花的人力物力，使新产品更早交付给用户，并能保持系列产品操作上的一致性，节省用户的培训学习时间，从而赢得用户的支持，使产品占有更广阔的市场。

8 用户界面层构件

用户界面层构件是一个软件系统的最终用户了解和使用系统最直接的窗口，它向最终用户传递系统的处理结果信息。用户界面层构件的外观表现形式、其内部组成构件的布局直接影响到最终用户使用系统的工作效率。界面的颜色选择，输入框和按钮的位置应当在整个系统中保持一致。不提倡在界面中使用过多的颜色，如果一个界面象调色板一样，会让用户感到无所适从。据统计报告显示，用户比较容易接受浅色调的界面。窗口中输入框和按钮在每个界面中的位置应当相对固定，不能一个窗口中的退出按钮在右下角，另一个窗口中却挪到了左边，这样会使用户感觉整个系统的操作没有一致性，给用户带来不必要的麻烦。

另外界面设计最好向用户提供个性化设置的选项，让用户可以按自己的意愿来调配界面。当前手机、彩电、音响等电子产品行业流行的‘彩壳随心换’在软件产品界面设计方面引起了巨大的反响。很多界面小巧的软件都提供了界面选择功能。大量媒体播放软件的界面选择是这方面的成功典范。

当前可视化开发环境中提供了大量的基本界面构件，开发新的用户界面层构件可以着眼于将基本构件加以组合、改进，从而提供更大粒度上的界面构件。对不同应用领域加以调查分析，提出每个领域的基准界面框架，从而开发面向特定领域的界面构件。

界面构件的设计应当由专门的界面装潢师来完成，因为他们了解怎样的色彩搭配符合用户的视觉心理，怎样的界面布局可以提高用户的工作效率。在实际中，很多极为普通的系统实施方案和最终系统被用户欢天喜地地接受的原因就是有一个令他们兴奋的界面。相反，有许多技术上很优秀的系统却因最终用户不喜欢而告吹。人们都相信产品包装和广告是神奇的[10]。

9 层次化软件构造的应用

在开发大型项目的过程中，不断地使用层次化软件构造的思想来指导项目的设计开发。在开发过程中和系统提交给用户后的维护过程中，笔者体验到了层次化带来的种种好处。在开发过程中，由于系统的底层构件早就开发好了，当上层要添加新的功能时，只须在相关下层构件的基础上用少量编码工作先开发一个新的特定实现层构件，最后和相应的界面构件集成起来就行了。这时得到的是用少量的时间就完成了一个新系统功能的高质量实现。此时的开发工作不再是困难重重的艰辛工作，而是类似搭积木那样的简单活动。在系统维护过程中，当发现问题时，能根据系统的层次结构快速准确地定位问题所在，在极短的时间内排除系统故障，从而保证用户业务操作的顺利进行。

上海市教育考试院网上招生管理系统是负责上海地区每年十万左右考生档案流动的控制系统。系统的成败关系到广大考生是否能进入高等院校继续深造，这个系统决定着考生的命运，有着重要的社会影响。该系统分成招办端、院校端和网络传输三个部分。招办端主要根据考生的档案信息按照投档规则将考生的电子档案传送到各个高校，当高校退档或录取考生时，招办端要进行严格的审核，决定是否退档或录取考生；院校端系统与招办端类似，它主要负责学校与校内各院系之间的投档、退档、录取控制工作；网络传输部分用来在招办和高校之间传输考生的档案信息和退档、录取等审核信息。在该系统的设计开发过程中，采用了层次化构造方法，将整个系统进行先水平后垂直的层次化分解，最后采用构件技术把各个模块开发成相应构件。当底层的比如网络传输操作、控制考生状态的操作、控制院校各专业招生计划人数的操作等都封装进构件后，高层的投档控制、退档审核、录取审核等操作的实现就像搭积木一样将下层操作构件组合起来，很快就完成了系统总体框架内容的填充。由于网上招生工作实时性很强，每一个阶段都有大量的统计，以便于招办控制招生政策，还有一些阶段性的录取信息需要及时向新闻媒体发布。通过开发的数据库报表构件，这些信息就可以用用户需要的形式提交给用户。有了层次化的设计，因为系统总体框架清晰明确，底层设施齐备后，就知道一个新功能需要哪些底层构件的支持，在上层添加新的功能就变得十分容易。这种层次化的设计大大减轻了系统维护负担，当系统出现错误时，根据系统的层次结构很快就能界定错误范围，并及时加以修正，从而确保了网上招生工作按时圆满地完成。

（下转 232 页）

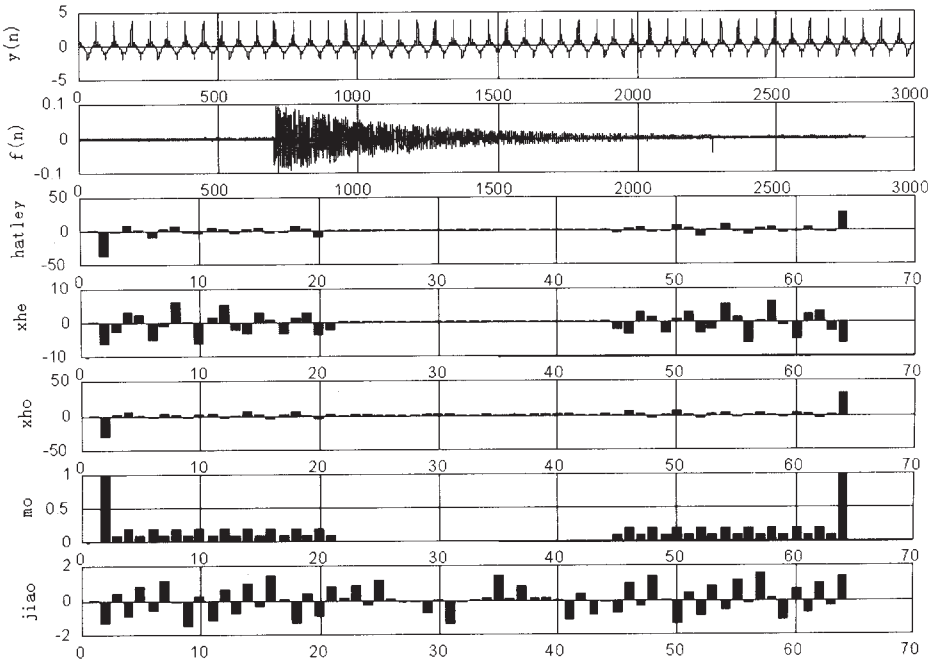

图 2　哈特莱变换仿真结果

4　结语

文章叙述了哈特莱变换原理及其在谐波分析中的应用。由于哈特莱变换的核函数是实函数不需要计算虚部，而且离散傅氏变换与哈特莱变换之间存在着确定关系，他们有许多相似性质，哈特莱变换象离散傅氏变换一样，也有快速算法，若求出了信号的哈特莱变换，再经过简单运算就可以求出离散傅氏变换，哈特莱变换用于谐波分析时可以节约运算时间和存储单元，提高运算精度。用硬件 DSP 芯片实现电力谐波分析时，回避了计算实部和虚部的诸多不便，因此较其他变换更适合硬件实现，具有经济、快速和高精度的优点。

（收稿日期：2002 年 12 月）

参考文献

1.胡广书.数字信号处理－理论、算法与实现[M].北京：清华大学出版社，1997
2.楼顺天，李博菡.基于 MATLAB 的系统分析与设计－信号处理[M].西安：西安电子科技大学，2000
3.李芬华，潘立冬，常铁原.电力信号频谱分析方法的设计与仿真[J].电测与仪表，38（429）：22

（上接 138 页）

由于网上招生工作责任重大，以前的系统在招生录取期间需要 6-8 个人来维护，采用层次化的构造以后，2001 年在现场维护的工作人员减少到 2 人。这不但减轻了客户的经济负担，也为开发商节省了大量人力物力。

在网上招生系统中开发的网络传输构件具有很大的通用性，在后继开发的需要网络传输的新项目中直接引入这些构件，大大节省了开发时间，并确保了新系统中网络数据的高效、准确传输。

10　结束语

该文针对面向对象和可视化开发环境中出现的系统结构混乱、开发维护负担重的情况，结合计算机网络中的层次参考模型和软件工程中的系统分解方法，提出了层次化软件构造的思想。通过当前流行的构件技术将层次化软件构造思想应用于实际的项目开发过程中，使软件复用的思想有了更具体的诠释。文章在对软件系统大粒度角度把握的基础上，将软件系统的组成构件划分为物理操作层构件、通用算法层构件、特定实现层构件和用户界面层构件四大类。最后用具体的项目开发实例说明了层次化软件构造方法可以使软件系统结构更加清晰，设计开发更加容易，可以大幅度减轻系统维护负担，并且系统开发过程中产生的构件可以在后继项目开发中得到复用，节省系统开发时间，确保向用户提供高质量的软件产品。

（收稿日期：2002 年 1 月）

参考文献

1.D Souza D，Wills A C.Object，Components and Frameworks with UML：The Catalysis Approach.Addison Wesley，Reading，MA，1999
2.Bertrand Meyer.Object-Oriented Software Construction[M].2nd Edition，Prentice Hall，1997
3.Roger S Pressman.Software Engineering，A Practioner´s Approach[M].4th Edition，McGraw-Hill，1997
4.Andrew S Tanenbaum.Computer Networks[M].3rd Edition，Printice Hall，1996
5.Mary Shaw，David Garlan.Software Architecture，Perspectives on an Emerging Discipline[M].Printice Hall，1996
6.Mark Priestley.Practical Object-Oriented Design with UML.McGraw-Hill，2000
7.张海藩.软件工程导论[M].北京：清华大学出版社，1998
8.郑人杰等.实用软件工程[M].第二版，北京：清华大学出版社，1997
9.周之英.现代软件工程[M].新技术篇，北京：科学出版社，2001
10.朱扬勇等.客户/服务器数据库应用开发[M].上海：复旦大学出版社，1997