第33卷 第4期 计算机研究与发展 Vol. 33 No. 4
1996年 4月 COMPUTER RESEARCH & DEVELOPMENT Apr. 1996

系统软件开发过程中的软件工程技术

王 健 程 虎
（中国科学院软件研究所 北京 100080）

摘 要 因为系统软件的复杂性和特殊性，如何控制和管理系统软件的开发过程是直接关系到开发成败的重要问题。本文认为系统软件不同于应用软件，有其自身的特点和规律。现有的应用软件开发方法和软件工程技术不能完全满足系统软件开发的需要。本文结合一个编译系统开发的实例介绍了系统软件开发过程中的一点经验和体会。

关键词 系统软件，软件工程，软件开发。

SOFTWARE ENGINEERING TECHNIQUES IN THE DEVELOPMENT OF SYSTEM SOFTWARE

Wang Jian and Cheng Hu
Institute of Software, Chinese Academy of Sciences, Beijing 100080

Abstract As system software is a complicated system, the management in developing system software is an important issue essential to the whole project. We argue that the system software is different from applicative software, and has its own properties. Current software engineering techniques can't satisfactorily meet the requirements of system software development. In this paper, some experiences on the development of system software are presented through analysing an example of compiler development.

Keywords System software, software engineering, software development.

1 问题的提出

软件这个词的范围比较广，它并不仅仅指应用软件。“所谓软件开发学科是指研究各类软件创建、维护等一系列开发活动的学科，其主要内容包括理论、方法和技术三方面的开发知识。软件可划分为智能软件、系统软件和应用软件三个领域。由于不同领域的软件采用的开发知识不同，

原稿收到日期：1995-02-18；修改稿收到日期：1996-01-10。本文受国家八五攻关项目资助。王 健，硕士研究生，主要研究领域：程序设计语言、语言编译、软件工程等。程 虎，1960年毕业于北京大学数力系，研究员，室主任，主要从事程序设计语言、语言编译、软件工程、人工智能、神经网络等方面的研究工作。

从而构成各自相应的方法，应用软件的开发是属于软件工程的领域”[2]。

虽然系统软件和应用软件属于不同的领域，在广义的范围上，系统软件和应用软件都是计算机软件，二者之间有一定的共性。那么，多年来计算机科学在应用软件开发方法方面所取得的大量成果是否能够不加任何改变地用来指导系统软件的研究和开发呢?

2 系统软件不同于应用软件

在我们开发系统软件的实践中，发现系统软件的研究和开发与应用软件的开发是显著不同的两个领域。我们认为，软件开发方法这个词具有一定的误导性，它容易使人们将“软件”理解为应用软件、系统软件和智能软件的总和。而软件开发方法的精确意义是指应用软件开发方法。

2.1 系统软件的问题域是不可结构化的

软件设计的基本目标就是人们试图用抽象的方式将现实世界的问题转化成程序空间的解题程序，通过程序的执行，控制计算机获得问题的结果。图1可以很好地表示这一过程[7]。

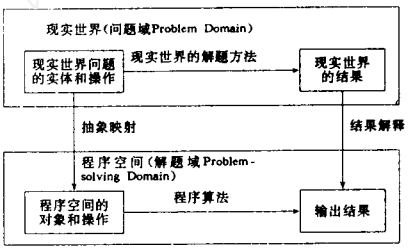

图 1

我们认为，系统软件的研制由系统软件的基础理论研究和程序开发活动两部分组成，而且理论研究成果直接关系到系统软件的成败和系统效率。有人说，系统软件如操作系统和编译程序等已经很成熟，现在开发这类程序只是工作量的问题了，实际上并不尽然。编译系统是计算机科学中发展很完善的一个领域，词法分析、语法分析、语义处理、代码优化、代码生成等在教科书上都可以找到完整的介绍，可是新的程序设计语言不断出现，相应地要求新的编译技术和新的程序设计概念的实现，而且旧有的一些方法和概念也需要进行修改。我们正在开发的 Fortran 90编译系统就是一个典型的例子，Fortran 90[5]中有一些新的程序设计概念如模块、超载、新的程序作用域规则，以及数据类型参数化、新的数据类型等，对这类机制的实现没有现成的技术和经验可循，在 Fortran 90之前的版本中没有出现过，因为该语言是九十年代制定的标准，上述特征是较新的程序设计概念，在其它语言中也比较少见。而且该编译程序的词法分析模块也不适于用传统的词法分析生成器来构造，因为 Fortran 90的词法规则不能用右线性方法来刻划，所以在我们的实现中仅为了完成词法分析工作就进行了多遍源程序的扫描，在程序设计语言的设计方面，非过程式程序设计风范的语言不断出现，它们都需要大量的理论为其建立语义模型和实现模型。再如操作系统，UNIX 经过多年的发展已经很成熟了，实际上 UNIX 一直没有停止它发展的脚步，CMU 的学者在1986年之后提出的微内核 Mach 操作系统正取代传统 UNIX 而风靡世界。所以一定要重视基础工作在系统软件研制中的重要性。

但是系统软件理论方面的一些研究成果难以使用软件工程技术提供的规范化工具来表达，其根本原因在于软件工程技术提供的工具描述能力差。例如研制一种程序设计语言的编译系统，像开发其它软件一样，第一步是进行需求分析，分析并描述程序设计语言的语义，这一阶段的成果是语言规格定义说明书。但是程序设计语言的语义既不能使用程序流程控制图，也不能用程序数据流向图这些工具来描述，甚至很难用形式化的数学方法(如公理语义、指称语义、操作语义)来

描述语言的语义，这些语义理论只是找到了一种数学系统与语言之间的解释关系，这种关系可以用作语言的理论基础，却不适合编译系统开发的需要。虽然国际标准化组织在程序设计语言的标准化方面做了很多工作，但为了描述程序设计语言的精确语义，在权威的国际标准中采用的仍然是BNF范式与自然语言的结合，从某种意义上说这只是一种权宜之计，因为找不到更好、更精确的描述手段，从Fortran 90国际标准文本的制订过程可以清楚地看到这一点。在标准文本正式发行之后很短的一段时间内就收到了对文本的100多个问题，其中有些是因为没有正确理解文本的内容，而有的问题确实是文本本身的缺陷造成的。现在ISO/ANSI还在组织会议征求意见，积极进行修改国际标准文本的工作。造成这种情况的原因不外乎如下两点：自然语言的二义性以及语言成份本身难以定义。在C++的标准化过程中也有类似的情况，ANSI/ISO C++委员会曾让一些形式化定义专家向各界解释他们的技术和工具，并介绍他们的观点，把真正形式化方案扩充到C++的定义中，这将有助于其标准化方面的工作。对此C++的提出者Bjarne Stroustrup认为：以该技术当前的水平，一个语言的形式定义若与某一种形式化方法同时设计，除了少数形式化方面的专家，是超出一般人的能力之外的。

所以，系统软件的问题域在现有的软件工程技术的意义上是不可结构化的。

2.2　系统软件的复杂性更高

从2.1节关于系统软件问题域的讨论中可以看出，系统软件涉及的层次比较多，有的是深层次理论问题，还有系统实现方面的工作，因此系统的复杂程度更高了。

分析系统软件的内部结构可以发现，模块之间的耦合性即依赖性非常强，对系统软件的某一个模块进行修改常常会影响到整个系统。软件封装、实现程序的模块化这个软件工程标准的愿望很好，但对系统软件很难实现，有的时候即使能够实现也要经过一个漫长的历程。最典型的一个例子就是UNIX操作系统，最初的UNIX操作系统各模块之间的调用关系过于复杂，依赖性太强，虽然大家早已认识到这一点，因为该系统的复杂性一直难以弥补这个缺陷。直到Mach操作系统的出现解决了这个问题，其间经过了近20年的时间。

2.3　系统软件对软件研制中管理技术的要求

既然系统软件的问题域更加复杂，如何管理和控制整个开发进程的问题就更加突出了，对研制过程中的人员管理、文档材料的管理等也有不同的要求。

软件工程就是指用工程化规范的方法来控制和管理软件的，从需求分析、设计、编码、测试到维护的生命周期的全过程。根据上面的分析，我们认为“工程”一词本身不能概括系统软件的全部内容，特别是理论方面的工作。系统软件的研究和开发方面还有新的课题需要进行研究，不能说这方面的内容已经成熟到用“工程的方法、标准规范和原则”等来指导各种系统软件的设计和实现。从这一点上讲，应用现有的软件工程方法来应付系统软件研究和开发之中的一切问题是不恰当的。

3　传统软件开发方法不能满足系统软件开发的需求

经过多年的发展和积累，已经形成了多种软件开发方法和理论，比较典型和著名的方法有SASD方法、JSP和JSD方法，快速原型法(prototyping)、OO方法等。这些方法中最古老同时也是最成熟、应用最广的是SASD方法，最新、最引人注目的是面向对象软件开发方法。

下面分别简单介绍一下最早的SASD方法和最新的OO方法，并以编译系统为例分析它们在系统软件研制中的应用。我们希望解决一个问题：所谓用工程化的方法来开发软件，其核心概念

是规范二字，从需求分析到最终系统的运行和整个软件开发过程都要受这些规范的约束，但是，软件开发方法提供的表示方法和固定的执行步骤能够处理系统软件开发中复杂而且常常需要灵活处理的问题吗?

3.1 SASD 方法与系统软件的研制

结构化分析方法(structured analysis)简称 SA 方法，是面向数据流进行需求分析的方法。结构化分析方法适合于数据处理类软件的需求分析。由于用图形来表达需求，显得清晰简明，避免了冗长、重复、难于阅读和修改等缺点，易于学习和掌握。结构化分析方法使用了以下几个工具：判定表、判定树、数据流图、数据词典和结构化语言。其中数据流图用以表达系统内数据的运动情况。数据字典定义系统中的数据。结构化语言、判定表和判定树都是用以描述数据流的加工。

结构化设计方法(structured design)简称 SD 方法，是由自顶向下进行软件系统的总体设计思想发展而成的。该方法使得数据流图可以向结构图进行系统地转换，因而可以和需求分析阶段采用的结构化分析方法很好地衔接起来。步骤：

(1) 研究、分析以及审查数据流图；
(2) 根据数据流图决定问题的类型；
(3) 由数据流图推导出初始程序结构图；
(4) 改进初始结构图；
(5) 修改和补充数据词典；
(6) 制定测试计划。

SASD 方法为计算机科学的发展做出了卓越的贡献，但是它并不适用于系统软件的开发，在我们研制编译程序时强烈地感到了这一点。

用正则表达式、右线性文法来描述程序设计语言的词法规则，BNF 范式来描述程序设计语言的语法规则，正则表达式、右线性文法、BNF 范式具有严格的形式理论基础-Chomsky 形式语言理论。所有这些表达均难以用判定树、数据流图和数据字典等形象但是表达能力极其有限的工具来代替。形式化表示方法不仅仅是直接面向程序设计语言实现的，而且已经有一整套的工具和方法来直接将面向语言实现的表示转换为可执行的程序，例如 YACC、Lex 以及自动编译生成系统。

符号表是编译程序实现中的一个重要工具。符号表的实现体现了语言的作用域规则的实现。在我们所做的编译程序设计中，首先用自然语言介绍了符号表查表算法的思想，说明了该算法能够满足语言的特殊要求，然后用伪代码将算法表示出来，最后使用 C 语言实现了符号表查表算法。对算法如果只使用 SASD 方法提供的程序框图的形式来表达它的设计思想，即使能够表示，该算法的可读性也一定极差，恰恰违反了软件工程的要求。使用程序框图的目的是为了使人们更容易理解程序设计思想和程序代码，但在系统软件开发中，根据 SASD 方法的要求只使用程序框图却起到了相反的作用。

目标语言结构的设计在编译系统实现中是个重要问题。如何保证目标语言与源语言之间的语义等价，对所选择方案的说明和解释在设计说明书中是必要的内容，而这也不是 SASD 方法的表示工具所能够胜任的。

我们认为，在系统软件研制过程中应该大胆使用结构化程序设计的思想和结构化程序设计风格等有益的内容，但是 SASD 方法的表示工具和工作步骤不适于系统软件的研制。

3.2 OO 方法与系统软件的研制

OO 方法近来在信息产业界炙手可热，甚至被认为是解决软件危机的一种希望。那么该方法是否也能为系统软件的开发带来新的希望呢?

3.2.1　P. Coad 和 E. Yourdon 方法简介[13]

Coad 和 Yourdon 方法是一种比较成熟的 OO 开发方法，它面向对象开发的整个周期。在分析阶段 OOA 将系统分成五个层次：主题、类和对象、结构、属性、服务，OOA 的任务就是通过分析问题域，建立系统的概念模型。OOD 在 OOA 模型的基础之上，除了原有的五个水平层次之外，将系统结构在纵向上划分为四个部分：人机交互部件、问题域部件、任务管理部件和数据管理部件，这样 OOA 和 OOD 的模型框架见图2。

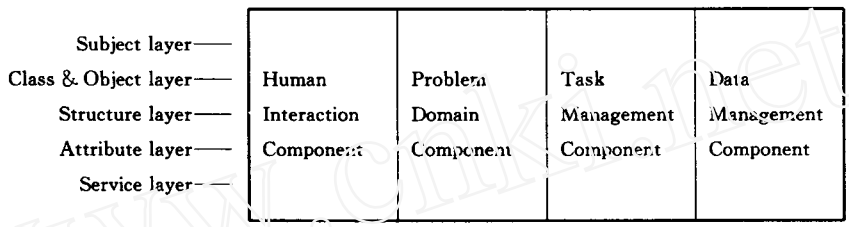

图　2

可见与其它软件设计方法一样，OO 方法的基本形式是给出基本的概念和原则，以及体现这些原则的可操作的表示方法和执行步骤。

对象概念是否适于描述语言各层次的语义，OO 方法的设计步骤能够很好地体现编译算法的执行吗？

3.2.2　OO 方法在编译程序中的应用

已经有人试图将 OO 概念引进编译程序的设计之中。OSU(Oregon State University)提出了使用多风范的编译程序模型[12]，其中包括了面向对象的概念，见图3。

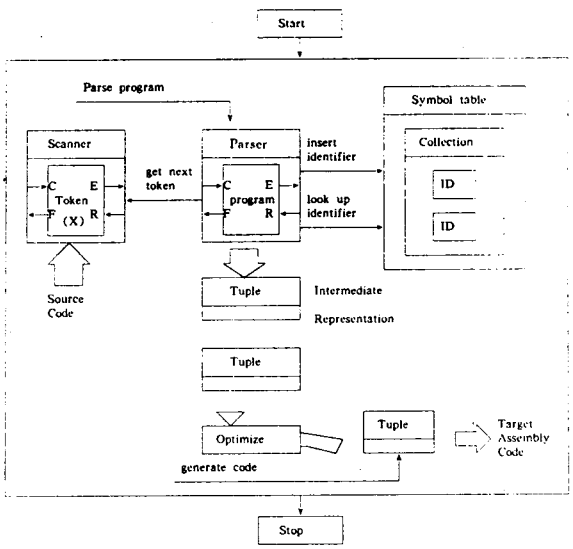

图　3

C:called;　E:Exists;　R:Retried;　F:Fails。

编译程序被看作一组相互作用的构件(agent)，这些构件包括有扫描器、分析器、符号表、元

组、优化器和代码生成构件。构件之间的相互调用过程就是把源代码翻译成目标汇编代码的过程。

扫描器的状态和行为是以逻辑程序设计风范的方式实现的，语言词法的逻辑规则为：数字的定义，字母的定义，两条扩展的词法规则：常量的定义，标识符的定义。消息“获取下一个符号”的作用是调用扫描器分析一个符号。源语言的词法规则用逻辑关系式来刻划，逻辑关系式属于逻辑程序设计的范畴。

源语言的语法是一组定义合法程序结构的规则。构件“分析器”用逻辑关系的方式封装语法规则，并在这些纯句法规则之上增加了语义规则。对扫描器的调用结果是把从源文件中抽取的词法记号转换为断言。传给分析器的消息“分析程序”被解释为一个断言，该断言查询源文件中的符号串是否满足源语言的语法和语义规则。特定规则的满足表示相应的源语言结构的出现，而一个结构的识别生成新的断言。源语言的语法规则使用逻辑关系式来刻划，逻辑关系式属于逻辑程序设计的范畴。

符号表接受分析器的请求把一个新的标识符和它们的属性插入到符号表中，或者在符号表中查找符号的属性，这类查找的解释依赖于标识符的作用域。该符号表的实现采用了面向对象程序设计风范。根据我们的经验，语言的作用域规则对符号表的数据结构和填表、查表算法影响很大，设计符号表的时候最重要的是根据语义规则设计出算法和符号表的结构，至于用什么语言来实现该算法并不很重要。如果一种语言的作用域单位规则发生变化，符号表的结构和查询算法也要修改，因为修改前后的符号表的共同之处不多，使得面向对象方法的继承性显示不了其优越性。当然OO语言的封装设施很完善，它以语言语法的形式约束程序设计人员按照高质量的程序设计风格来开发程序，不过任何有经验的程序设计人员会自觉地遵守软件工程准则的约束。

在识别源语言语句的过程中，分析器生成程序的中间表示元组构件，子类区分了元组的状态和行为，例如ALU代表一条算术逻辑指令。函数optimize对元组流进行处理，把其中的某些元组流替换成高效的代码，每一个转换流中的元组发出一条“生成代码”的消息来生成目标汇编代码。

上面的模型也包括了过程式程序设计风范，模型的初始和结束部分由过程式语言来完成，在该模型中过程式程序设计风范起到了一种“粘合剂”的作用。

更技术性的工作有关于Lex，YACC和面向对象语言作为基础语言的研究[11]，Lex和YACC的基语言是C，即Lex和YACC只能与C一起使用，[11]中介绍了自动编译工具Lex和YACC与面向对象语言C++一起使用的方法和界面，并利用C++建立了语法树，提供了一个遍历语法树的例子。

显然，这些工作证明了OO技术能够用到编译程序的构造之中，但是没有对生成的编译程序的效率和规模等做分析和比较，所以不能证明OO技术是编译程序最好的实现方法。OSU的模型只是对编译程序的符号表的设计与实现应用了面向对象技术，而不是整个编译程序，对编译程序的其它环节，OO技术很难取代传统工具的地位。OSU模型中的每一个构件(agent)与传统编译系统模型中的阶段(phase)之间有显然的对应关系，实际上多风范模型只是对编译的各阶段进行了封装，并用新的名字分别为每个阶段重新命名，各阶段之间的调用方式使用消息传递来进行而已。我们认为，传统的编译系统模型结构清晰，模块调用关系清楚，功能划分以及模块化程度都很好，OSU模型的提出无论在概念上还是在实现上并不比传统模型更为优秀。

仔细分析OSU模型的实现，使用的仍然是原来的概念和思想，只是表述形式概念的实现语言换成了Leda，可以用图4来表示OSU模型的实现过程。大家知道，在图灵计算的意义上，任何

一种完备的计算机语言都是等价的，只是在描述具体问题上有的语言比较方便，而有的不方便。既然两段程序基于的设计思想是相同的，用什么语言来表示并不是重要的事情。而且使用OSU模型中间多了一个对传统编译概念建模的过程，效果反而不好。当然，[12]中工作的出发点是证明多风范语言Leda的应用，所以基于同样的概念，它证明可以使用不同的语言来表示，形象地说这种做法是“新瓶装旧酒”，问题是对于编译程序而言，旧瓶本来就很漂亮的呀！

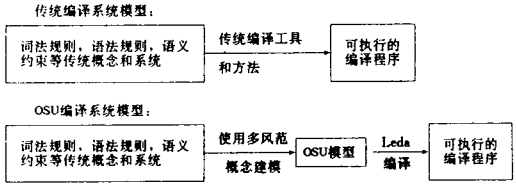

图　4

3.2.3　OO软件开发方法不适于系统软件开发的原因

传统软件开发方法虽然问题的描述从程序的物理实现，即语言层次提高到整个程序甚至软件的结构层次，提出结构化设计的思想，但是传统的软件开发模型未能提供对软件开发的一个基本要素——客观世界对象结构的认识，致使实际开发中从客观世界到机器程序空间的映射完全凭人们的经验和直觉进行，建立的逻辑模型也不是面向客观世界对象结构而是面向程序解空间的。

与传统软件开发方法相比，OO方法在对现实世界建模方面有着明显的优势。有人认为，OO的最基本优点不是编程技术的改进，而是分析问题域的视角的变化，人类认识客观世界的过程往往并不是首先对系统的功能进行理解，而是集中于对所要解决问题空间的理解，及对解题系统所处的环境和应用领域的理解，理解是通过实体的识别，然后再识别实体服务或功能的方式来进行的。人类认识世界的过程中普遍采用了三个构造方法：区分对象和属性，区分整体对象及其组成部分，不同对象类的形成和区分。面向对象的软件开发从识别客观世界的实体出发，建立系统的概念模型，并以此为基础进行软件设计和编码。由于面向对象方法面向客观世界，根据稳定的对象建立系统模型，因而这些模型往往是稳定的，可以很好地适应需求的变化，可理解性、可扩充性和易维护性均很好。

可见，OO的最大特点是能够以较小的语义差距来模拟现实世界中的过程，但是在系统软件开发领域，OO的该特点反而成为了缺点。系统软件需要描述的对象不是现实世界中的客观存在，而是由人构造出来的一个形式系统或者虚拟机器，所以OO不适于解决系统软件中的问题。既然OO方法与它的问题域之间也有语义差距，它与传统的SASD方法相比，就没有任何优势了，我们认为这一点上说面向对象的软件开发方法反而成为了面向程序解空间的，这是该方法不能适于开发编译程序的根本原因。

3.3　测试计划

任何一项工作对最后的成果都要进行鉴定验收。鉴定的方式有两种，科技成果鉴定和软件确认测试、软件的规格测试。

技术鉴定会审查材料是听取研制报告、技术报告、测试报告和用户意见。技术鉴定会的重点是评价技术的可行性与水平的先进性。

近年来，人们逐渐认识到技术鉴定会的形式不能够完成对软件产品的鉴定任务。国家科委于1994年10月26日以宋健主任令的形式，发布了新的《科学技术成果鉴定办法》。1995年1月1日实施，根据鉴定办法，软件产品大多数是由企事业单位自行开发的一般应用成果、再经商品化而推出的，即从1995年起软件产品不再由政府科技管理部门组织同行专家进行鉴定，而是直接投放市场。软件产品可通过用户使用认可，也可以通过第三方的产品检测机构确认测试来认证。

系统软件不同于一般的软件产品，应该采取技术鉴定会的方式还是软件确认测试和软件规格测试的方式呢?

在作者参加的一次系统软件鉴定会上，鉴定专家之一指出为什么不按照规范的测试程序来验收工作成果呢? 要求鉴定者回答得很妙：我们做的是对提出的想法的有效性考核，而不是进行软件产品的确认测试和规格测试。

作者同意这种观点，无论是科技产品鉴定还是由第三方主持的软件产品确认测试和软件规格测试，都不能满足系统软件研制的要求。我们认为对于系统软件既要组织有效性考核即科技成果鉴定，同时也要进行软件产品确认鉴定测试。为了保证系统的质量，不仅要进行最终结果的验收，而且在项目的实施过程中也要进行阶段性的检查。

4 我们的对策与经验

我们负责了国家"八五"计划中编译系统开发的任务，并具体承担 Fortran 90编译系统的研制，该项目的实施严格遵循软件工程方法的要求，这里介绍一下我们的体会。

系统软件的研制既包含了理论性的色彩，同时又要求得出一个具有鲁棒性的(robust)实用系统，而这两个要求相互之间有一定的矛盾性。

我们从系统软件研制的特殊性出发，总结各种开发方法的合理部分，力求形成一套相对合理的开发方法。在研制过程中，我们主张不受具体的软件工程的方法、生命周期、概念和模型的限制。这一点我们同 Budd 等人的想法是相同的，即针对问题领域的特点采用不同的策略[12]：

All the research is concerned with providing the "right" set of constructs for the programmer, allowing the programmer to use more than one mode of thinking (Paradigm) for complex problems, but the approach varies widely from project to project.

区别是 Budd 等人只关心程序设计风范，我们是从软件工程的角度来谈这个问题的，其范围包括了软件开发技术、日常工作的管理、人员管理、软件文档管理等各个方面的问题。开发过程的每一阶段及每一个问题都可以寻找最佳的解决方案，然后把它们组合起来，形成自己的工程控制方案。

对软件开发方法，我们使用裁剪的手段，对所有能够帮助理解设计思想的工具都采取拿来主义的态度。例如采用了 SASD 流程图来形象地表示算法的执行过程，部分功能模块的设计使用了原型法。

软件不仅指程序代码，还包括数据和文档。对于软件的文档国家技术监督局发布了一系列的国家标准，有关的有：GB 8566-88《计算机软件开发规范》、GB 8567-88《计算机软件产品开发文件编制指南》、GB 9385-88《计算机软件需求说明编制指南》、GB 9386-88《计算机软件测试文件编制规范》、GB/T 12504-90《计算机软件质量保证计划规范》、GB/T《计算机软件配置管理计划规范》。课题的技术组根据国家标准又作了具体的要求。

任何措施或方法都需要落实到日常管理工作之中，两个重要的办法是举办开发讨论会和定期填写规范的工作总结报告。走查(walkthrough)是软件测试中的概念，我们鼓励开发组成员在开发讨论会上相互对系统的设计思想和实现进行走查。

人员管理是软件管理工作的一部分，软件工程的目标之一是以规范化的形式记录成果，这样不会担心开发人员中途离开项目的开发工作，即使部分开发人员离开，继任者也能够顺利地继续进行下去。但是系统软件的研制不仅需要编制程序的经验，而且需要一些基础性的理论知识，基

础性的知识不是短时间内可以获得的。所以我们提倡开发人员从一而终，同时在选择开发人员时严格控制质量关。

我们认为在选择合理的开发方法的时候，应该重视原型法的思想在系统软件开发中的作用。C++的设计实现过程实际上就是一种快速原型开发，“从没有什么C++的书面设计，设计、编写文档和实现是同时进行的。自然C++的前端是用C++编写的，从未有过什么“C++计划”或者“C++设计委员会”。在整个期间，C++都是在解决用户遇到的问题中，通过作者和他的朋友和同事之间的讨论而不断进化的。“,”只是在C++成为已确立的语言之后，更为传统的组织机构才出现了。但即使在那时，我仍然对参考手册负责，并对什么人应加入到其中拥有最后的决定权，直到1990初整项任务被移交给ANSI C++委员会为止”[9]。

总之，“Right method at right time”。

5 结　论

本文首先提出系统软件与应用软件是不同的两个领域，然后具体地以SASD和OO方法在编译系统中的应用为例说明了应用软件开发方法不适用于系统软件的开发，最后介绍了我们工作的一点经验。

系统软件种类繁多，通晓所有系统软件的知识几乎是不现实的。因为作者知识范围所限，对操作系统、数据库系统、网络系统未有涉及，本文得出的一些结论很可能失之偏颇，敬请读者批评指正。这项工作显然还需要做大量的探索、研究、试验和实践，才能形成一套完整的方案。看了本文之后，我们相信在读者研制和开发系统软件特别是编译系统时，会注意其中的特殊性质，想出有效的相应措施，在设计系统软件时会有好处。

参 考 文 献

[1] ISO/IEC 1539, Fortran 90, 1991, Gevenea, Switzerland.

[2] 仲萃豪，丁茂顺，孙洪生，叶　农. 应用软件开发方法. 计算机科学，1991，(1)：5－14.

[3] 孙玉方. 文件系统质量保证技术与实施. 中软公司质量保证讨论会，1994年2月.

[4] 金茂忠. 软件产品质量的保证——软件测试技术. 计算机世界报，1993年10月20日.

[5] 程虎，王健等. Fortran 90语言规格定义说明书. 中国科学院软件研究所技术报告，1993年7月.

[6] 程虎，王健等. Fortran 90编译系统概要设计和详细设计说明书. 中国科学院软件研究所技术报告，1994年7月.

[7] 李京. 面向对象的软件构造技术研究[中国科技大学博士论文]，1993年5月.

[8] 黄民德. 产品经测试，用户更放心. 中国计算机报，No. 2，1995年1月15日. 13.

[9] Bjarne Stroustrup. C++的历史：1979～1991. 计算机世界报，1994年11月2日. 99－125.

[10] 汪成为，郑小军，彭木昌. 面向对象的分析、设计及应用. 北京：国防工业出版社，1992年9月.

[11] Bruce Hahne, Hiroyuki Sato. Using YACC and Lex with C++. *ACM SIGPLAN Notices*, December, 1994, 29(12).

[12] Justice T P, Pandey R K, Budd T A. A multiparadigm approach to compiler construction. *ACM SIGPLAN Notices*, September, 1994, 29(9).

[13] Peter Coad, Edward Yourdon. *Object-oriented Analysis*. Yourdon Press, 1990.

[14] Dyadkin L J. Multibox Parsers. *ACM SIGPLAN Notices*, 1994, 19(7).

[15] Aho A V, Sethi R, Ullman J D. *Compilers, Principles, Techniques, and Tools*. Addison-Wesley Publishing Company, Reading, Mass, 1986.