新世纪高职高专信息管理与信息技术专业教材

竹宇光　刘兰娟　编著

软件开发方法

RUANJIANKAIFAFANGFA

上海财经大学出版社

新世纪高职高专信息管理与信息技术专业教材

软件开发方法

竹宇光　刘兰娟　编著

上海财经大学出版社

图书在版编目(CIP)数据

软件开发方法/竹宇光,刘兰娟编著. —上海:上海财经大学出版社,2001.7

新世纪高职高专信息管理与信息技术专业教材

ISBN 7-81049-597-6/TP·10

Ⅰ.软… Ⅱ.①竹… ②刘… Ⅲ.软件开发-高等学校:技术学校-教材 Ⅳ.TP311.52

中国版本图书馆 CIP 数据核字(2001)第 041683 号

RUANJIAN KAIFA FANGFA

软 件 开 发 方 法

竹宇光 刘兰娟 编著

责任编辑 麻俊生 封面设计 周卫民

上海财经大学出版社出版发行

(上海市中山北一路 369 号 邮编 200083)

网 址:http://www.sufep.com

电子邮件:webmaster@sufep.com

全国新华书店经销

上海崇明晨光印刷厂印刷装订

2001 年 9 月第 1 版 2001 年 9 月第 1 次印刷

787mm×960mm 1/16 12.5 印张 266 千字

印数:0 001—4 000 定价:21.00 元

总 序

高职高专是我国高等教育的重要组成部分。大力发展高职高专教育，培养大批社会急需的各类应用型专门人才，对于提高我国劳动者素质、建设社会主义精神文明、促进社会进步和经济建设，都将起到重要作用。

按照教育部关于高职高专人才的培养目标，构建适用的教材体系是高职高专教育发展的重要环节。经过编辑委员会、作者和出版社的共同努力，《新世纪高职高专系列教材》将陆续出版，我向他们表示诚挚的祝贺和感谢。

综观这套系列教材，具有以下明显的特点：

——充分体现了教育思想的新成果和新观念，贯彻前瞻性原则，注重提高学生思想道德素质、文化素质、业务素质和社会责任感，突出创新精神和实践能力培养的要求。

——体现了应用型、复合型、外向型人才的培养目标和规格定位。

——坚持整体优化原则，注意处理好高职高专教材与本科教材的区别，做好各层次知识的互相区分和衔接。

——处理好理论教学和实践教学的关系，使教材更贴近实践，注重培养学生的操作能力。

当然，高职高专教学是一项系统工程，在编好教材的基础上，要真正抓好教学工作，还必须在运用教材过程中辅之以其他的配套措施，尤其注重以下几点：

首先，牢牢把握“以培养高等技术应用型专门人才为根本任务，基础理论知识适度，技术应用能力强，知识面较宽，素质高”的高职高专教育特色。

其次，引入现代教育技术，积极实行启发式、讨论式教育，激发学生学习的主动性，培养学生的务实精神与创新意识。

再次，要特别加强教学与实践相结合，培养学生的动手能力。

由于我国高等职业教育还是新生事物，起步不久，本系列教材不可避免地存在一些问题。殷切希望读者能随时向编写人员提出意见，使之进一步完善，更加适应高职高专教育发展的需要。

储敏伟

2001 年 2 月

目 录

总序 …… (1)

第一章 基础知识 …… (1)
第一节 软件的概述 …… (1)
一、软件的概念 …… (1)
二、软件的特征 …… (2)
三、软件与硬件的联系和区别 …… (2)
第二节 软件的发展 …… (4)
第三节 软件开发的原则 …… (8)
一、通用的原则 …… (8)
二、具体的原则 …… (14)
第四节 软件的生命周期 …… (16)
一、软件的生命周期 …… (16)
二、软件的生命周期模型 …… (18)
本章小结 …… (22)
复习思考题 …… (22)

第二章 软件开发的规划与可行性分析 …… (23)
第一节 软件开发规划的内容 …… (23)
一、软件开发规划的任务与特点 …… (23)
二、软件开发的战略规划 …… (25)
第二节 软件开发的可行性研究 …… (27)
一、经济上的可行性研究 …… (28)
二、技术上的可行性研究 …… (31)
三、人员的可行性研究 …… (31)

四、社会的可行性研究 …… (32)
五、可行性研究报告 …… (32)
本章小结 …… (33)
复习思考题 …… (33)

第三章 软件的需求分析 …… (34)
第一节 软件需求分析的原则、方法 …… (34)
一、软件需求分析的工具 …… (35)
二、分析的其他工具 …… (49)
三、需求分析的原则 …… (52)
第二节 需求分析的任务 …… (53)
一、软件需求分析的任务 …… (54)
二、软件需求分析的过程 …… (56)
第三节 验证软件需求 …… (67)
一、验证软件的正确性 …… (68)
二、验证需求的方法 …… (68)
本章小结 …… (69)
复习思考题 …… (69)

第四章 软件的总体设计 …… (70)
第一节 软件设计目标和任务 …… (71)
一、软件设计的目标 …… (71)
二、软件设计的任务 …… (71)
第二节 总体设计的过程 …… (74)
一、挑选最佳解决方案 …… (74)
二、确定软件的结构 …… (75)
三、设计文档 …… (76)
第三节 软件总体设计的内容 …… (77)
一、处理过程的抽象 …… (77)
二、信息隐蔽和局部化 …… (78)
三、模块化 …… (78)
四、模块的独立性 …… (80)
第四节 从 DFD 导出结构图 …… (91)
一、结构图 …… (91)

二、从数据流图导出结构图…… (93)
本章小结…… (103)
复习思考题…… (103)

第五章 软件的详细设计…… (104)
第一节 软件详细设计的概念…… (104)
第二节 结构化设计方法…… (105)
第三节 详细设计的工具…… (108)
一、程序流程图 …… (108)
二、盒图 …… (109)
三、问题分析图 …… (110)
四、判定表 …… (112)
五、过程设计语言 …… (114)
第四节 Jackson 设计方法 …… (120)
一、Jackson 方法概述 …… (120)
二、Jackson 设计方法 …… (120)
本章小结…… (129)
复习思考题…… (129)

第六章 软件的测试与调试…… (130)
第一节 软件的测试…… (130)
一、软件测试的基本概念 …… (130)
二、软件的测试 …… (135)
第二节 软件的调试…… (144)
一、软件的调试步骤 …… (144)
二、软件的调试方法 …… (146)
三、软件的调试策略 …… (147)
本章小结…… (149)
复习思考题…… (149)

第七章 软件的维护…… (150)
第一节 软件维护的概念…… (150)
一、软件维护的定义 …… (150)
二、影响软件维护工作量的因素 …… (151)

三、软件维护的策略 …… (152)
四、软件维护的代价 …… (153)
第二节 软件的维护过程 …… (154)
一、维护的组织机构 …… (154)
二、软件维护的申请报告 …… (154)
三、软件维护的工作流程 …… (155)
四、保存维护记录 …… (156)
五、评价维护活动 …… (157)
本章小结 …… (157)
复习思考题 …… (158)

第八章 面向对象的软件开发 …… (159)
第一节 面向对象方法的产生 …… (159)
一、传统开发方法的不足 …… (159)
二、传统开发方法存在不足的原因 …… (161)
第二节 面向对象技术的概念 …… (163)
一、对象 …… (163)
二、类 …… (165)
三、消息和多态性 …… (165)
四、继承 …… (166)
第三节 面向对象的开发方法 …… (167)
一、Rumbaugh 方法 …… (167)
二、Booch 方法概述 …… (177)
三、Coad 与 Yourdon 方法 …… (181)
本章小结 …… (188)
复习思考题 …… (188)

第一章　基础知识

软件是计算机系统的重要组成部分。本章通过软件的发展、开发软件所要遵循的原则、软件的生命周期等方面介绍软件的概念。

第一节　软件的概述

一、软件的概念

计算机系统由硬件和软件两大部分组成。

软件由计算机程序和与之相关的各种文档资料组成。

计算机程序是软件的一部分，程序是为了达到一个预期的目的而编写的计算机指令集合。如要解决某一问题或是进行某一项控制而编制的一系列执行指令，这些指令之间具有先后顺序，计算机程序必须要以某种形式的计算机语言（如高级语言 Pascal，C＋＋等）来表达所要执行的每一步骤。

程序的各种文档资料也是软件的组成部分。它之所以成为软件的一部分，是因为没有相关的文档资料，使用人员就无法正确使用软件、维护软件，至于对软件的更新、改造等更是无从谈起。如同一台新的家电设备，买来后，如果随机没有携带有关的使用说明，那么用户就无法正确地使用，有些功能可能不知道如何去操作。所以，软件还应包括有关程序的各种文档。

软件作为一种比较特殊的产品，除了人们日常所说的计算机程序以外，还应包括该软件的各种文档资料，两者缺一不可。文档资料随着软件的不同会各不相同，它是在软件的开发过程中产生的，并在使用过程中起到指南或手册的作用，而在日常维护中，如软件需要更新、改造时文档资料可以起到支持和帮助的作用。因此，只有计算机程序，而没有相关的文档资料是不完整的。程序的文档资料是多方面的，如程序的设计说明资料、技术资料、使用说明等等。

当然，有些程序的文档资料和软件已经融为一体。这类软件的某些文档必须在准确

地安装了软件并启动以后才能供使用者使用。其前提就是该软件必须正常运行，否则就如同你购买了一台技术上非常先进，且只带一盘使用说明录像带的录像机，而你又不知道如何启动它，那么录像机的性能再好、功能再全、技术再先进对你来说都无济于事，因为你无法看到它的使用说明书。

二、软件的特征

软件与硬件不同，它具有如下的一些特征：

(1)软件具有“非实物”的特性。软件是一种逻辑性系统元素，不是物理性系统元素。为了完成一个软件系统，需要建立庞大的逻辑体系，其中包含了人们大量的智慧和许多创造性的劳动，但这些都是无形的，而最终所表现出来的形态就是计算机的程序和相关的各种文档。有了程序，有了文档，人们还无法看到其性能，对它所能完成的功能也无法了解。只有当该软件在计算机上被从头至尾执行了，人们才能检验其是否达到了预期的功能。即使是软件在计算机中执行时，人们对它还是“看不见、摸不着”。所以软件具有“非实物”的特性。

(2)软件具有“不磨损”的特性。软件必须由计算机来执行才能体现它的使用价值。只要该软件稳定、可靠，且对它的要求始终没有变化，那么在任何时刻被执行时，它的使用价值都不会改变，而且每次执行所发挥出来的功效是一样的，决不会像其他的机器设备那样，随着时间的推移其效率会逐步下降。当然，实际工作中，尽管软件不会磨损，但它会“退化”。这是由于对它的要求提高了，或是执行它的环境发生了变化，原来的软件其功能和性能需要完善和提高，而不是由于长时间的使用被“磨损”了。

(3)软件只能设计出来，不能制造出来。因为软件是人类智慧的结晶，是人们创造性劳动的产物，所以它的出现并不需要经过任何形式的工艺加工和生产。一旦它被设计出来，那么在任何形式或者环境下的拷贝，其质量是一样的，决不会出现质量参差不齐的情况。当然，为了提高软件的开发效率，研究和开发人员借鉴了制造业、建筑业的管理方式，建立了以软件的规划、分析、实施、测试等为主要内容的“软件工厂”，实现了软件开发的自动化或者半自动化的过程。因此，在“软件工厂”中，只要设计完成，并在计算机上执行后达到了人们的预期目标，该软件就可成为“工厂”的“产品”——软件产品“出厂”了。所以，软件只能被设计出来，而不能用传统意义上的制造进行生产。

三、软件与硬件的联系和区别

不论是大型或巨型计算机，还是便携式或掌上计算机，任何一个完整的计算机系统均由硬件和软件两部分构成。两者既有联系，又有区别，有机地统一在一起。

计算机的硬件是“看得见、摸得着”的物理实体，如键盘、驱动器、显示器、中央处理器等等，但这些设备必须要靠软件才能“动”起来。那么，何时“动”？怎么“动”？要协调这些

次序也需要靠软件。在软件的大家庭中，有一类就是专门从事对计算机设备及相关资源进行控制和管理的软件，即所谓的操作系统 OS(Operating System)。确实，有一部分软件的编制必须紧紧地依赖于计算机的硬件，甚至在开发这些软件时，开发人员必须对相关硬件的各种物理特性了如指掌，如各种计算机系统的驱动程序、基本的输入输出程序 BIOS 等等。任何一台计算机在运行各种软件之前首先要运行操作系统。如果没有类似于操作系统的软件，而只有计算机的硬件，那这台计算机纯粹是一件摆设，没有任何的使用价值。同样，没有基本的计算机硬件，而只有软件，也无法发挥出软件所能完成的功能。因此，硬件和软件是相辅相成、缺一不可的。

但是，计算机系统中的硬件和软件还是有比较明显的区别。

首先，软件具有“非实物”的属性，而硬件恰恰相反，它就是实实在在的物理设备。因为计算机的任何硬件设备都是由超大规模的集成电路通过有机地组合而构成的。

其次，与软件所具有的“不磨损”特性相对，硬件是会“磨损”的，或是含“老化”的。因为硬件是物理设备，那么随着时间的推移，硬件的故障率会开始上升，这往往是由于遭受了灰尘或不洁空气的污染、震动、使用不当、经常工作于极限温度等许多不良环境的积累而引起的，从而导致硬件的“磨损”。但是硬件磨损的故障不会导致软件出现故障，即不会使软件“磨损”。图 1—1 和图 1—2 分别表示了硬件和软件在其生命周期中的故障率的情况。在图 1—2 中，软件的故障只是在初始阶段比较频繁。但随着时间推移，对软件不断改进和完善，可以使软件的故障率保持在一个比较低的水平。

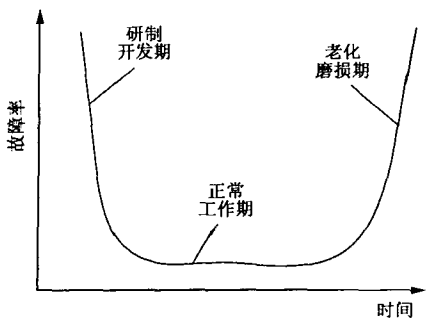

图 1—1　硬件故障率曲线

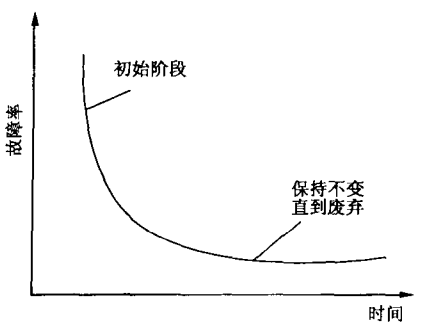

图 1—2　软件故障率曲线

再则，软件是设计出来，而硬件除了设计以外，还需要运用有关的工艺来生产制造。所以软件的质量完全由设计过程来控制，而硬件的质量除了要在设计过程中进行控制外，生产过程的改变或是工艺上的差异也左右着硬件的质量。如果硬件存在质量问题很可能要遭到报废，而软件的质量问题可以通过将实现过程中的错误予以纠正来解决，因而在质量保证方面，软件比硬件具有更大的灵活性。

第四，硬件一旦出现故障或磨损后，可以用相同的备用元器件进行更换，从而使硬件

系统能很快得以恢复，转入正常的工作状态。但软件却没有相同的备用“元件器”可言，因为软件出现的故障，不外乎是这么两种情况：要么是设计考虑不周造成的，要么是编程时候造成的。不论是哪一种情况造成的软件错误，都使得软件没有备用“元器件”可供更换。所以在对计算机系统进行维护时，软件要比硬件复杂得多。

第二节 软件的发展

自 1946 年世界上第一台电子计算机诞生至今只有半个世纪多一点，而计算机技术的发展却是空前的。人类发明了计算机，但对飞速发展的计算机技术又感到难以置信。在 20 世纪中，在硬件方面，40～50 年代是电子管时代，60 年代是晶体管时代，70 年代是中大规模及超大规模集成电路时代，80 年代网络开始蔓延并开始渗透至人类生活的各个领域，90 年代计算机技术、网络技术和通信技术的结合形成了信息高速公路。

伴随着硬件的发展，计算机的软件及其开发技术在观念及目标等方面也发生了很大的变化。

1. 40～50 年代

由于每一台计算机的指令系统都是单独设计的，没有规律，所以组成计算机指令的“0”、“1”序列让人难以辨别、难以记忆，再加上当时计算机的价格昂贵，于是懂计算机的人很少，会使用计算机的人只能是凤毛麟角的极少数几个了。当时，美国全国的程序员也总共只有 500 人左右，普通人是不敢问津的。那些能够写出让计算机完成某些计算任务程序的人员往往被认为是逻辑天才。当时计算机的用途主要是科学计算或是军事上有关的计算问题，对程序的要求是能执行、速度快、能产生出可以接受的结果，并希望能用最少的资源来获得最大的运算能力。因此程序的质量完全依赖于程序员个人的技巧。

随着计算机的体系结构日趋成熟，计算机的生产已经可以形成批量的规模，价格也开始下降，于是其应用范围很快地扩大了。由于应用范围的扩大，对程序的要求也提高了。当在原来用“0”和“1”组成的机器语言程序中加入一条新指令时，程序员必须亲自对整个程序进行检查，以确定所有相关的指令和操作经过变动后仍然是正确的。当计算机的应用变得越来越复杂时，编写程序也变得越来越困难，但同时人们渐渐地发现机器指令和存储体的地址可以用一些便于记忆的符号来代替，这样有利于程序员熟记计算机的全部指令，于是出现了用指令符号来编制程序的办法。在此基础上，进一步发展就出现了汇编语言。由于汇编语言是依赖于机器硬件的，因而称之为“面向机器的语言”。虽然“0”、“1”数码换成了符号，但它们之间有严格的一一对应关系，因此人们开发出了专门的程序，用来将程序员写的符号语言程序翻译成机器语言程序（也叫目标程序）。这种程序称为“汇编程序”，并成为当时计算机中必不可少的软件。

使用汇编语言编写软件，必须要了解机器的某些细节，如累加器、寄存器的个数以及

它们之间的关系、内存的地址等等，因为汇编语言还是依赖于计算机的。正是由于针对硬件的这个特点，使得汇编语言所编写的程序和用机器语言所编写的程序一样，都是短小精悍、效率高、执行速度快的程序，这是因为这些程序充分利用了计算机硬件的特性。时至今日，在一些计算机公司中仍然在使用汇编语言来编写系统软件，以保证高质量软件的功效。

与机器语言相比，汇编语言已经有了一定的改进，但还是面向机器的语言，这给计算机的应用普及和普通百姓学习、使用计算机带来了许多困难，因为编写程序必须要了解或熟悉计算机的某些细节。随着程序内部控制的日趋复杂，到了 50 年代末，人们发现按照某种规则来组织、描述程序的助记符有助于对程序的理解，特别是用数学符号来表示的数学运算比单纯的用汇编语言所描述的更容易理解。这一发现导致了一系列早期的高级语言开始出现，而利用高级语言和各种数据类型能够比较容易地编制一些复杂控制的程序。

在计算机出现以前，人们以高昂的代价设计各种专用的硬件来解决一些特定的问题，如计算、存储等等。计算机的出现带来了各种各样的存储介质，可以将大量的数据信息存储在这些存储介质上，还可以被反复利用，同时运算速度也大幅度地提高。技术上的这一突破，使计算机能够代替那些昂贵的硬件和逻辑功能。既然计算机的存储介质上可以存放数据信息，那么所编制的软件程序也能够存储在这些介质上，这就是冯·诺依曼提出的存储程序概念。存储的程序可以被反复地利用和执行，从而可以解决那些较大规模的问题。这是那一阶段技术的另一个突破。

2. 60 年代

软件的开发主要使用各种符号语言，包括汇编语言和面向应用的高级语言。由于高级语言有一定的“语法规则”，编制时比较接近人的思考习惯，又不需要顾及计算机的内部构造和不同机器的特点，所以编程人员可以集中精力考虑解决问题的步骤。编写出的程序通过编译程序的检查和翻译，计算机就能执行。这给编程工作带来了极大的方便，也大大提高了编程速度，扩大了解决问题的规模和难度。

这一时期软件开发具有这些特点：一是编程的随意性很大，要看懂和理解别人所编制的程序非常困难，因为编程完全依赖于编程人员的才能和技巧；二是开始把一组小程序链接以后组成一个大程序，以完成单一的、综合的系统功能，如 IBM 公司花费 1 亿美元、用了 6 000 人年开发出了操作系统 OS 360。计算机的应用已经不再局限于计算问题，其应用范围扩大了，开始进入商业、银行等领域。同时开发了一批规模庞大的软件。由于应用面的扩大，软件开发人员不仅为自己的研究工作编制程序，更主要的是为用户，特别是那些对计算机一无所知的用户能更好地利用计算机的功能而编制程序。同时软件的开发人员也开始意识到了软件，特别是大型软件存在着代价高、错误多、质量低、维护难的情况。

这一阶段的主要发展是各种高级语言的出现，而且编程可以独立于计算机本身的结构和指令系统，从而使编程工作不再由专职的程序设计人员来担任，这为计算机应用领域

的进一步扩大起了很强的促进作用。同时各种高级语言的出现也使相应的编译程序得到了极大的发展。此外，计算机在非数值计算领域方面的能力也得到了广泛的认可，从而进一步拓展了计算机的应用领域。只是在开发这些应用软件时，开发人员与用户之间的交流和联系还停留在通过程序内部写上精确的注解以及相关的规格说明书来进行。

由于这一时期软件开发的随意性，因此难以保证软件的质量和开发的成功，常常是大量的人力、物力和财力的投入，最终换来的却是失败。问题是软件的失败事先很难预料，而一旦失败往往是难以挽回的。许多有识之士对此进行了探讨和研究。荷兰科学家 E. W. Dijkstra 就指出：不应该简单地只考虑编写程序，就期望产生一个正确的结果，应考虑如何把软件进行划分。他提出了结构化程序设计方法，在程序设计中取消 GOTO 语句，以提高程序的可读性为目标。后来，Jacopini 等人证明了只要有“顺序”、“选择”、“循环”三种基本的控制结构就可以实现任何单入口、单出口的程序。这个程序结构定理就是结构化程序设计的理论基础。只使用三种基本控制结构的高级程序设计语言，以及只有一个入口和一个出口的程序设计原则，共同形成了新的程序设计思想、方法和风格，使程序变得清晰、易读、易修改。这就是结构化的编程方法。

3. 70 年代

这一时期的各种技术方案的主要特点都是提供可见的管理对象，并启用调查统计和分析工作来度量工作成效。在软件的开发方法上，一方面数据结构和算法从程序中分离出来，成为了一种独立的研究对象，使有关人员和学者可以集中精力对其进行研究并加以改进，并形成了相关的理论。利用数据结构和算法的理论，可以优化对计算机中的存储设备与运算控制设备这两项资源的利用。另一方面，结构化程序设计方法进一步发展成了结构化开发方法，包括结构化分析方法和结构化设计方法。此外，软件的开发虽然主要还是要靠个人的努力，但对复杂问题的求解，开发人员已经意识到更应强调的是程序的清晰，而不应过分地强调所用的技巧。

在这一时期，计算机的软件系统分成了两大阵营：系统软件和应用软件。同时，为了解决大型系统的问题，发展了数据独立的概念，即数据可以文件或数据库的形式存在，而与系统相脱离，同时也发展了并发访问的程序执行机制。于是开发人员不仅要考虑系统的功能，而且还要考虑系统的状态。对大型系统来说，重要的不仅仅只是源程序或可执行程序，那些复杂的系统规格说明也变得相当的重要，因为这种规格说明中包括了系统的功能、性能、可靠性及输入/输出要求等等。

这一时期的理论成就是数据结构，算法理论也得到了很大的发展，可以用数理逻辑、断言等方法对程序正确性进行推理，开始采用建设工程管理的方法来进行软件开发，提出了各种软件开发的模型，形成了软件工程。同时人们也认识到一个完整的软件系统，既包含可执行的程序系统，还应包含独立的数据结构和程序系统的规格说明书。

4. 80 年代

由于软件系统规模的不断扩大，开发一个软件系统的重点已经由单纯地编程转向了对系统的构造方法，即开发一个软件系统不再强调编程问题，而强调开发时的一些与系统有关的问题，如系统各部分之间的接口及把各部分集成为一体的技术问题等等，同时对复杂的文档资料及规格说明的管理成为管理系统结构的重要手段。这一时期，大量的数据库应用系统出现了，使得各种商业事务及社会生活等领域的各方面问题的描述可以转化为文字、数字，并以数据的形式进入了计算机及数据库，形成了各种类型的计算机信息系统。

这一时期，数据库应用系统大量出现，特别是关系型数据库得到了很大的发展。关系数据库的客户/服务器 C/S(Client/Server)理论成为这一时期的重大突破。关系数据库把诸如数据定义、数据浏览、查询处理等都用同一形式来表示，而且运算简短、清晰。在 C/S 的体系结构中，客户端主要负责获取数据信息的输入及将有关的查询处理结果表示成输出结果的形式；服务器端则负责数据的存储、处理，并对客户端来的请求给出回答。二者层次分明，各司其职，互不干扰，不但极大地方便了并行操作，而且对流水处理或并行操作也提供很好的支持。系统中可以广泛地采用非过程性的查询 SQL 语句，使查询的处理速度得到提高。同时与关系数据库相应的第四代语言的出现，图形用户界面的出现，关系数据库与电子表格的相容等等，都使广大用户能够非常容易地参与软件的开发工作。此外个人计算机 PC 的出现，使得网络、分布式系统也得到了很大的发展。

5. 90 年代

在 90 年代，处理的不仅仅是文字、数据符号，还有多媒体，同时也是计算机网络大发展的年代。Internet 的技术逐渐走向成熟，并得到了广泛的开发和利用，而网络浏览器和 Java 的诞生，使人们看到了 Internet 蕴涵了巨大的商业价值。因此，各个软件开发商，无论是开发操作系统的，还是开发数据库管理系统的，或是开发其他应用软件的，都纷纷推出了自己关于支持 Internet 的新产品。这一时期的许多软件，不仅提高了个人的生产率，而且还通过支持跨地区、跨部门、跨时间的群体共享信息、协同工作来提高群体和集团的整体生产率。几千万用户通过 Internet，跨时空地直接、并发交流信息，共享资源。因此使用的突发性成为网络应用的一个特点。

Internet/Intranet 普遍受到人们的欢迎，其技术以人们难以想象的速度迅速得到发展。电子邮件、远程教育、远程医疗、电子商务以及基于 WEB 和浏览器的应用极其迅速地得到普及。由于 Internet 是一个遍布全世界的巨大计算机网络，而整个网络中的各节点所用的平台不可能完全一致，于是程序在整个 Internet 上的可移植性、互操作性就成为软件开发者必须要考虑的一个重要问题。而 Java 语言的诞生正是满足了这个要求。该语言有着良好的网上移植性、安全性，且在编程上也较简单。

随着时间的推移，软件的开发技术还在不断地发展。回顾软件的发展过程，我们应该认识到：

(1)软件技术变动大,新技术不断出现。各种软件技术的应用必须根据所要解决问题的特点来选择。

(2)软件不是自然界的产物,它是人类思维的创造物,也是人类智慧的结晶。

第三节 软件开发的原则

所谓"原则"就是工作中的一些基本事实、规则及有关的规定。它是人们实践经验的总结,可以看作是工作中的真理或方法。在进行软件开发过程中,不论采用了何种技术、工具或语言,这些原则都必须遵循。换言之,原则是超越具体技术、工具、语言之上的,是普遍适用的准则。对于不同的工作,会有不同的要求,那么工作中也就必须遵循相应的原则。有些原则是通用的,而有些则是不同开发阶段的具体原则。

一、通用的原则

通用的原则是在整个软件开发过程中从头至尾都必须遵循的,它将在开发的每一个阶段中得到体现,而且它不仅能在大型系统中用来处理复杂的事物,即使是小型系统也可以参照执行。

1. 严格化的原则

任何一个大工程、大项目的开发,都有严格的要求和规范,软件开发与工程项目的开发一样。遵循严格化的原则,可以达到开发出正确、成功的软件产品的目的。虽然软件开发过程是一个创造性的工作过程,既不精确,也不准确,常常是一时灵感的产物。但严格化并不会抑制创造性,相反遵循严格化的原则,可以达到开发的目的,保证所开发软件的质量。因为通过严格的分析和设计,可以增进对创造性成果的信心,从而促进创造性的工作。

严格化对软件产品的可靠性、可维护性、可移植性、可理解性等等都会产生一定的影响。例如,一份严格的、详尽的文档可以排除在软件开发过程中的二义性、不完整性,因而对提高开发效率,提高软件的质量都是十分重要的。同时对系统的维护,也是一个很重要的保证。

软件产品的开发过程可能各不相同,但如果能将所经历的不同步骤都详细地记录在案,那么管理人员就可以精确地掌握和控制整个软件的开发过程,及时地评估,适时地调整,进而提高软件开发的效率。

2. 分隔化的原则

分隔化的原则实际是人们在日常生活中解决难题的一种行之有效的方法。分隔化的原则要求人们在解决一个复杂问题时,应该从不同的方面把问题分隔开来,然后单独地加以考虑。这样人们可以集中精力去解决每一个方面的问题。例如,要解决计算机的处理