大学计算机专业教程

软件工程原理与应用

陈世鸿　朱福喜
黄水松　陈　磊　编著

武汉大学出版社

图书在版编目(CIP)数据

软件工程原理及应用/陈世鸿,朱福喜,黄水松,陈磊编著.—武汉:武汉大学出版社,2000.12
大学计算机专业教程
ISBN 7-307-03074-8

Ⅰ.软… Ⅱ.①陈… ②朱… ③黄… ④陈… Ⅲ.软件工程—教材
Ⅳ.TP311.5

中国版本图书馆 CIP 数据核字(2000)第 40779 号

责任编辑:毕卫东　　责任校对:程小宜　　版式设计:支　笛

出版:**武汉大学出版社**　(430072　武昌　珞珈山)
(电子邮件:wdp4@whu.edu.cn　网址:www.wdp.whu.edu.cn)
发行:新华书店湖北发行所
印刷:武汉大学出版社印刷总厂
开本:787×1092　1/16　　印张:20.375　　字数:459 千字
版次:2000 年 12 月第 1 版　　2000 年 12 月第 1 次印刷
ISBN 7-307-03074-8/TP・89　　定价:26.50 元

前　言

1968年在北大西洋公约组织的一次学术会议上，第一次使用了“软件工程”这个词，它标志着人们试图用“工程”的思想和“工程化”的实施方法来解决大型软件研制中面临的种种困难和混乱。软件工程的目标在于研究一套科学地开发软件的方法，并与此相适应，发展一套方便的工具系统，直至整体的开发环境，力求达到用较少的投资获得高质量软件的目的。

钱学森教授在“关于思维科学”一文中把各学科划分为“哲学、基础科学、技术科学、工程科学”四个层次，计算机软件工程属信息科学领域的工程技术层。因此，与其他课程的特点不同，软件工程是一门“设计”课程，强调实际训练，课堂教学只能讨论软件工程学中的一些基本概念、原则和方法，更重要的是，读者必须选用一种方法实施一个从计划、分析、设计、编码、到测试的软件开发全过程，以便从中得到实际的从事软件工程的训练和经验。

本书是一本简明实用教程，目的在于使学生对软件工程科学的全貌有个系统地了解，从而使学生的注意力从个人小程序的独立活动转移或扩大到大型软件开发方面。

全书共分15章。第1～2章，介绍软件的发展和软件工程的基本概念；第3～11章依据软件生产过程介绍了传统软件设计法，以及面向对象软件设计法在软件计划、需求分析、功能设计、软件编码、测试和维护中使用的模型、技术和具体实施步骤；第12、13章分别讨论软件开发过程中的维护与管理问题；第14章是一个小型项目采用软件工程方法开发的完整实例；第15章介绍了软件工程的重要支柱——软件开发环境，并讨论了软件开发环境研究的问题和发展方向。

本教材的主要特点是简明实用，它是在作者10多年来讲授软件工程课程的讲稿和主持参加多个大中型软件项目开发的实践经验基础上撰写的。重点介绍软件工程的各种开发方法，并强调读者的实践训练。因此，在附录B中列出的17个中小型项目可作为学生实践时选用。本教材可作为大专院校计算机专业软件工程课程的教材和教学参考书，也可作高级程序员、系统分析员的培训教材。对从事软件开发的工程技术人员，本教材可供自学和参考。

作者水平所限，谬误之处，敬请读者批评指正。

作　者

2000年5月于珞珈山

目 录

第1章 绪 论

1.1 软件及其发展的三个阶段

人们常说的计算机系统是由硬件和软件两大部分组成的。用计算机求解问题，程序是不可缺少的，程序称软件的实体部分，程序只有在硬件载体上运行才可获得所求问题的解，硬件和程序是求解问题的根本条件。然而，仅有程序也会给使用者带来诸多不便，好的程序应有相应的文字资料，如各种开发规格说明书、用户手册等，通常称这些文字资料为文档。文档不仅对使用者是必须的，而且对程序开发者也是至关重要的，特别是由多人经过多年才能完成研制开发任务的大型程序，人与人之间，开发者与使用者之间都需要有规范的书面文档规定程序的功能、使用环境、方式方法等。所以严格地说，程序和软件是两个不同的概念。程序是指源程序代码或机器可直接执行的程序代码，而软件是指程序加上开发，使用和维护该程序所需要的全部文档。

五十多年来，计算机科学工作者在计算机硬件和软件两大领域做了艰苦并卓有成效的工作。在这50年间，计算机硬件已经历了电子管、晶体管、集成电路和以大规模集成电路为主体的四代计算机，并正向大规模并行处理计算机和新一代智能计算机发展。每十年发生一次"脱胎换骨"的巨变，其发展速度是任何其他行业无法比拟的。硬件的迅猛发展也促进软件的不断发展和革新。软件的发展始终迎着程序设计自动化这一根本目标进军。经过软件工作者多年的努力，其发展也经历了高级程序设计语言兴起时期，结构化程序设计时期和软件工程与软件开发环境时期这三个主要阶段，并向自动化更高的智能化软件迈进。当然，软件的发展也不断向硬件提出新课题，二者是相辅相成的。

下面简单介绍软件发展三个时期的主要特点。

1. 程序设计语言兴起时期

计算机软件工作者围绕着"程序设计自动化"这个根本问题推动了软件的发展。早在20世纪50年代初期，为了改变机器代码编程的困难，有人提出用符号代替机器代码编程(汇编语言的萌芽)，随即出现了正规的汇编语言。1956年J. Backus设计并实现了第一个高级程序设计语言FORTRAN，开辟了程序设计自动化的新局面。FORTRAN语言的诞生兴起了软件工作者研究高级程序设计语言的热潮，并且不断追求程序设计语言的表述能力，把语言的表述能力看作是衡量程序设计自动化程度的高低。在这种动力的激发下，自50年代末期到60年代末期，世界各国的软件工作者研究了上千种高级程序设计语言，最著名并得到世界认可的几种主

要程序设计语言是FORTRAN、ALGOL、BASIC、APL、COBOL、PL/I、LISP等。与此同时，为了使高级程序设计语言编制的程序能在计算机上有效地运行，研制了大量与高级程序设计语言相适应的辅助软件系统，如编译系统、解释系统、连接装配程序、编辑程序等，并建立了形式语言、数据结构、编译原理等基本理论，逐步建立起计算机软件学科。

20世纪60年代末期，一方面由于提供了强有力的高级程序设计语言，使计算机的应用日益广泛，逐步渗透到各行各业乃至人们的日常生活中，开发的程序越来越庞大和复杂，但对大型复杂软件的开发又缺乏科学的方法作指导。另一方面，由于软件工作者过分追求语言的表述能力，使语言中引入了不甚合理、难于理解的成分。上述原因往往造成开发程序的极不可靠性，程序不可靠性成为60年代末期的突出问题。例如：IBM公司开发的OS/360系统，耗资几千万美元，花费了五千多人年，拖延了几年才交付使用，交付使用后每年发现近100个错误。OS/360系统开发负责人Brooks生动地描述了研制过程中的困难和混乱："……像巨兽陷入泥潭作垂死挣扎，挣扎得越猛，泥浆就沾得越多，最后没有一个野兽能逃脱淹没在泥潭中的命运……程序设计就像是这样的泥潭，一批批程序员在泥潭中挣扎……没有料到问题会这样棘手……"。甚至还有比OS/360更糟的软件系统，既花费了大量的人力财力，结果还是半途而废。

如1979年，美国财政部(GAO)对美国联邦政府开发的九个软件项目进行了调查，虽然工程规模并不大(九个项目的总值不足700万美元)，但结果令人十分忧虑(见图1.1)。47%的资金花费在从未使用过的软件上。更糟糕的是，另外29%的资金花费在那些交付后需再开发以及交付后在GAO调查时已被遗弃的软件上。总之，成功的项目仅占3%～4%。该调查虽已过20年，但软件开发的现状并没有得到根本性的改变。我国软件生产现状更是如此，除少数中小规模软件系统有成功的例子外，成功的大型软件开发实属罕见。因此，研究新的软件开发方法是软件产业的当务之急。

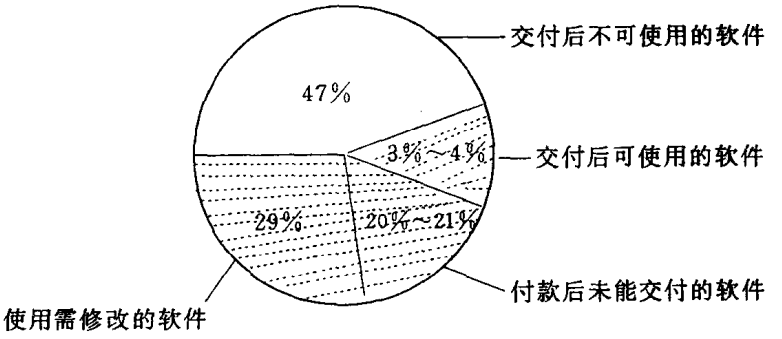

图1.1 成功与非成功软件项目比例

为了解决软件的可靠性问题，软件工作者又把精力放在程序设计方法和程序正确性证明

方法的研究上来。这就是下面要介绍的结构化程序设计时期。

2. 结构化程序设计时期

这一时期研究的核心问题是程序设计方法和程序正确性验证的方法。程序设计方法以 E. W. Dijkstra 于 1968 年提出的"结构化程序设计方法"为代表,引导人们广泛地研究如何能保证程序正确性的设计方法和验证方法。在当时近五年时间内就提出了模块化、结构化、由顶向下逐步求精、程序变换、程序的推理与综合、数据类型抽象、程序正确性证明、符号测试等各种程序设计和验证的方法,虽然有些方法至今仍作为理论上的探讨,但这一时期的研究成果使软件工作者深刻地认识到,没有科学的方法作指导,就不可能生产出高质量的软件产品。同时提出了以正确性、结构清晰、便于阅读、便于测试和维护为软件质量的标准。

另外,针对语言有些成分的不合理性,语言工作者对语言各成分逐一分析和批判,设计出了更简明、更适合写大型程序的高级程序设计语言,典型的有 PASCAL、MODULA 等。ADA 语言的出现是对命令型语言的一次科学总结,它汇集了各种语言中的优良成分,并在语言中引入了程序设计方法学的机制,为开发大型软件提供了良好的环境。最近几年面向对象的程序设计语言如 Smalltalk,C++、Visual Basic、Delpha、JAVA 等获得更广泛的应用,它们支撑新的软件开发方法——面向对象的软件开发方法。

程序设计和程序正确性验证方法的研究,使程序设计走向更科学的道路,在某种程度上提高了软件可靠性,但程序设计过程仍然采用低效率的手工方式,周期长、代价高,远远不适应软件生产的需要。因此,在 1968 年北大西洋公约组织的一次学术会议上,第一次创造了"软件工程"这个词,其宗旨是用工程化方法开发软件,以求解决软件生产效率低、代价高、可靠性差等问题。

3. 软件工程与软件开发环境时期

在传统的软件开发过程中,软件生产的各个阶段没有明显的界线和明确的分工,有的软件项目在未弄清其要求和功能的情况下,就动手编码,使得编出的程序结构混乱,难于测试,难于使用,花费巨大的财力和人力,得到的是不可靠软件,甚至有的软件不能使用而作废。受其他行业工程技术的影响,人们试图用工程的方法开发软件项目。因此,软件工程是研究按工程化方法开发软件(特别是大型软件)的技术和方法。同时,大型软件的开发应有良好的环境(与机器制造业要有工作母机一样),我们把支撑软件开发的软件系统集称为软件开发环境。软件工程与软件开发环境也是本书讨论的内容之一,在此不赘述。

当前,在研究大型软件开发新方法的同时,人工智能软件开发所需的语言、方法、技术和环境,以及软件体系结构的研究也引起软件工作者的极大兴趣。其研究成果除广泛用于人工智能软件(如专家系统)的开发外,还用于软件工程方法中,以开发出自动化程度更高更好的大型软件开发环境。

1.2 软件工程

软件工程是研究“大型”软件开发的技术、方法、工具、环境和管理工程的学科。首先，软件工程的重点是“大型”软件开发。假设一个人一个月能完成1 000行代码的设计，按此工作效率，一个10 000行代码的软件是否花十个月或者十个人花一个月能完成呢？回答是否定的，当软件规模从1 000行增加到10 000行时，虽原代码行增加了十倍，但是，软件复杂性程度的增加却远远超过十倍。

受其他工程学科的影响，考虑到研制一个软件系统同制造一台机器或建造一座大楼有许多相同之处，因此，人们参照其他工程学科的方法和技术进行软件的研制，试图用“工程化”的思想作指导来解决软件研制中面临的困难和混乱，从根本上解决软件危机。因此，人们提出的所谓软件工程是：应用一些本学科理论和工程上的技术来指导软件的开发，从而达到用较少的投资获得高质量软件的目的。

然而，正像其他工程学科一样，软件工程也有它自身应遵循的一套科学设计原理和方法。由于软件产品具有抽象性、逻辑性、非实物性等特点，其研制与维护过程本质上是一个“思考”的过程。因此，其设计原理和方法与其他工程学科有很大不同。

同时，由于软件工程是相当复杂的，涉及到的因素很多，所以不同软件项目使用的开发方法和技术也是不同的，而且有些项目的开发无现存技术，带有不同程度的试探性。一般说来，软件工程方法规定了：① 明确的工作步骤与技术；② 具体的文档格式；③ 明确的评价标准。本书将系统地讨论上述有关方面的内容。

与此同时，人们发现没有好的开发工具或环境，也会严重影响软件开发的效率和质量，研究开发大型软件的通用工具和环境也是软件工程学研究的重要内容之一，本书将介绍几个成功的软件开发工具与环境，供软件开发者参考。

总而言之，软件工程的目标在于研究一套科学的工程化方法，并与此相适应，发展一套方便的工具系统，力求用较少的投资获得高质量的软件。

1.3 软件工程面临的问题

发展至今，摆在软件工程学面前仍有许多方面的棘手问题，以下就软件产品的价格、可靠性、可维护性、生产效率和软件重用等方面作一些简单分析。

1.3.1 软件价格

由于新的电子元器件迅速发展，生产的自动化水平愈来愈高，计算机硬件的信息处理能力不断提高，特别是处理速度和存储量增加，质量越来越好，而成本却大幅度下降。与此相反，在

计算机硬、软件投资所占比例中，软件投资却在迅速上升，硬、软件投资比在1:3以上是很平常的事。

随着计算机应用新领域的不断出现，新的软件会愈来愈多。另一方面，计算机硬件不断更新换代，当新一代计算机取代现有旧的计算机时，软件系统也要随之更新。然而软件开发是高度密集型智力技术，生产过程中大量工作仍是手工劳动，所以远远跟不上应用和硬件发展的新要求，软件价格上升的势头将会持续下去。

1.3.2 软件可靠性

软件可靠性是指软件系统能否在既定的环境下运行并实现所期望的结果。通常花费在软件测试和排错的代价大约占软件开发代价的40%左右，即使如此，也不能保证经测试的软件就没有错误。好的程序设计技术，如模块化，结构程序设计和面向对象的设计等，能有助于减少程序设计中的错误，但是，测试是很难保证一个大的软件系统完全正确性的。Dijkstra对测试的效果讲了一句非常精辟的话："测试只能说明程序有错，而不能保证程序无错"。但至今为止，还没有保证程序正确性的更好的方法，所以说软件可靠性是软件工程面临的又一个难题。而且大的软件系统其测试是一件很复杂费时的事情，在大的软件系统中，程序的任何改变都可能会涉及到许多人，这就使得软件系统花在测试阶段的代价非常之大。为了提高软件的可靠性，就要付出相应的代价，重要的是在可靠性和测试代价之间作出权衡，在第11章的软件测试中，还要详细讨论这一问题。

除了软件的正确性之外，在更广泛的意义下，软件的可靠性还应该包括安全性和健壮性。

所谓安全性是指对于一组合理的输入，软件系统能给出正确的结果；而对于用户有意或无意的不合理输入，系统应能拒绝这种输入，并指出输入的不合理性，提醒用户注意。对于不合理的输入"系统束手无策"，甚至导致失败或系统被破坏，这种不安全的系统是不能投入使用的。

健壮性是指软件系统对环境的适应性。当软件系统所处的环境发生变化时，例如存储量不够、硬件故障等意外事件发生时，系统都能按照某种预定方式作适当处理，有效地控制事故的蔓延，不致丢失重要信息，从而避免灾难性的后果。

1.3.3 软件维护

尽管有人对软件维护有不同的看法，但修改与完善已经运行了的软件仍然是必要的，越需要反复维护的软件，生命力越强。因此，对软件产品需要花费很大精力做修正完善工作。我们把对已经运行的软件系统所需做的修正完善工作称为软件维护。在一软件开发机构中，往往要把很大精力花费在维护已有的软件上。为什么要花费很大精力进行软件维护呢？这是因为已运行的程序总是在变化，如功能的修改和增加，运行环境的变化，故障需排除、性能需提高等。另外还要对文档进行修改、补充和完善。除维护要花很大精力外，维护后可能由于副作用产生新问题，这又要花费精力去测试和解决。

因此，如何减少维护的总工作量，这也是软件工程面临和要解决的又一主要问题。

1.3.4 软件生产率

计算机的广泛应用使得软件的需求量迅速上升，世界各国普遍感到软件人力资源的缺乏，这种趋势仍在继续增长下去。但是，目前的软件生产基本上处于低效率的手工生产方式，而生产出来的软件的质量又很不理想，不适应市场对软件的大量需求。如何以较少的投资获得"高产优质"的软件，这是软件工程最主要的研究目标之一。其实现途径有二，即良好的软件工程设计方法和方便有效的软件设计工具与环境。

近十几年来，软件工程已研究发展了各种各样的软件工程化设计方法和技术，这些方法和技术的实用范围是各不相同的，有的方法适用于数据处理系统，而有的方法适用于实时控制系统。各种软件方法的风格也迥然不同，有的方法仅仅是一组指导性的原则，而有的方法则有较具体的设计原则，有的方法建立在严密的数学基础之上，而有的方法则是实际经验的总结。本教材将选择一些比较实用的、有代表性的方法和技术进行讨论。

方法和工具、环境之间有着密切的联系，相辅相成。在软件开发和维护的不同阶段，都应该有这样或那样的软件工具和环境，帮助开发人员自动地完成许多数据分析和处理工作。这样的工具环境愈完善，软件的设计和维护也就愈方便，这是当前提高软件生产率的一个很重要的方面。人们希望研究出一套系统的软件开发方法，一组成龙配套的软件开发环境，从而为软件人员提供一个能覆盖整个软件生存期的良好的工作环境，这样可使软件生产率大为提高，但要达到上述目标还有许多理论和技术问题有待解决。

1.3.5 软件重用

软件重用也是提高软件生产率、降低软件成本的一个重要方面。当人们一次又一次地去设计一些互相雷同的程序时，很多劳动都花费在一些基础重复的工作上，软件重用的技术旨在减少这种重复。

可重用的软件单位可以是程序代码，如子程序；也可以是软件逻辑结构和框架，如软件模块的详细设计表示；甚至可以是一种成熟的设计思想和原则。

软件重用方式大体上可分为两种。最简单易行的一种方式是直接组合可再重用的软件单位，称黑盒构件。这当然会有许多使用上的限制，用得最多的是各种程序库。另一种方式是按模式重用，有如下几种情形：

1. 面向问题编程　用面向问题的语言，或者非常高级程序语言(Very high－level Languages)编写程序，这样的程序实质上是问题的一种高级描述(Specifications)。再由此出发，可以在机器中实现多种不同的语言。

2. 程序变换　可以设计这样的源代码变换系统，使之能把一种语言的源程序变换成另一种语言的程序。例如，FORTRAN 语言有其丰富的应用程序库，只要设计成功从 FORTRAN

语言到另一种语言，比如说PASCAL语言的变换系统，就立即可以得到PASCAL语言的同样丰富的程序库。

3. 程序系统生成 许多程序系统虽然表面上看来很不相同，但是它们的基本结构模式却是一样的。例如，面向语言的结构化编辑器，除了语言的结构框架或者模式不同之外，编辑器其他部分的功能基本上是相同的。另外，高级程序设计语言编译中的语法分析和代码生成也是如此。这些事例表明可以设计出一个关于一定的结构模式的系统，给出一些特定的具体说明和要求，这样系统就能产生出特定的软件来。通常所说的许多程序自动生成系统，就是这种情形的软件重用的例子，而近几年提出的软件体系结构是这类基本结构模式的进一步抽象。

与软件重用密切相关联的另外两个方面是软件移植和规范化的软件构件。软件重用是近年来非常受人们重视的一个新课题，预料将会有很强的生命力。

以上列出的是软件工程所面临的几个重要问题，可以说，这些问题均未获得圆满地解决，有待进一步的发展和提高，这也是不断推出软件开发新方法和技术的根本所在。

1.4 软件的生存期

在1.2节中已指出，软件工程是按工程化的方式开发和管理软件，其突出特点是把软件的开发过程分为不同阶段，软件系统的生产过程通常分为如下六个阶段。

1. 软件计划 在设计任务确立前，首先要进行调研和可行性研究，理解工作范围和所花代价，作出软件计划。

2. 软件需求分析 通过调查来具体分析用户要求，并用规格说明书表达出来，规格说明书中通常包括功能需求、性能需求、环境的要求与限制等内容，该文档作为用户与软件开发人员之间相互共同的约定。

3. 软件设计 设计阶段分为总体设计和详细设计。总体设计决定系统的结构，并给出各部分的相互调用关系，相互间传送的数据结构以及每个部分的功能说明；详细设计则要设计出每一部分的内部实现算法，不同的设计技术有不同的系统结构。

4. 软件编码 编码阶段是根据任务的特点，选择合适的语言与相应支持环境，按软件设计说明书的要求为每一部分编写出程序代码。

5. 软件测试 测试的任务是发现和排除程序中存在的错误，测试步骤通常分单位测试和联合测试，经过测试和排错，得到可运行的软件。

6. 软件维护 维护是指对已交付运行的软件继续进行排错、修改和扩充。在1.3.3中已讨论过维护的有关问题。

上述六个阶段称为软件生存期。前五个阶段称软件开发阶段，第六个阶段为软件的使用和维护阶段。在软件开发期中，测试阶段的工作量最大。通过对多个大型软件系统的调查分析，证实前五个阶段所占工作量的百分比如图1.2所示，而开发阶段与维护阶段所占百分比如图