软件工程技术

主　编　王立峰　刘宝祥
副主编　方宁德　张仲选　杨　慧

RUANJIANGONGCHENGJISHU

吉林大学出版社

软件工程技术

主　编：王立峰　刘宝祥
副主编：方宁德　张仲选　杨　慧

吉林大学出版社

内 容 简 介

本书主要讲述软件工程和实践的概念、原则及一些典型的技术和管理方法。全书共十章，第一、二章简述了软件工程的历史，问题和有关术语的定义；第三章提出了管理软件开发过程的基本原则；第四、七、八、九章分别对软件生命周期中的重要阶段：用户要求说明、设计、实现和质量控制等阶段进行了讨论；第五、六和十章偏重于软件开发的管理、估算和教育问题。

本书可以作为计算机系的研究生和本科生学习软件工程的教科书，也可以作为从事软件工程的各方面人员：用户、管理、专业技术人员、程序员等的参考书。

软件工程技术

主　编：王立峰　刘宝祥

副主编：方宁德　张仲选　杨　慧

吉林大学出版社出版　　吉林大学出版社发行
（长春市解放大路85号）　　吉林工业大学印刷厂印刷

开本：787×1092毫米 1/16　　1989年11月第1版
印张：11.75　　1989年11月第1次印刷
字数：270千字　　印数：1—1500册

ISBN 7—5601—0367—7/TP·3　　工本费：6.00元

前　言

软件工程是当前计算机科学的一个极为重要的领域。电子计算机在短短的40多年历史中取得了较为迅速的发展。但是，随着软件的规模和复杂性的增大，应用范围的日益增广，软件的价格在急剧上升，在许多以计算机为基础的系统中，软件是花钱最多的项目。当软件系统越来越大时，质量也越来越成问题，开发的进程也越来越难以控制。由于没有能力生产能够利用80年代处理机的巨大能力的高质量软件，计算技术的进展也受到了限制。这就是人们谈论的软件危机。软件工程正是针对这种危机而发展起来的一门学科。

本书的目的是为软件工程过程涉及的各方面人员提供关于软件工程和实践的综合性指导。全书共分成十章。第一至第三章，简述了软件工程的历史和当前存在的问题，给出了软件工程和所采用的术语的详细定义，提出了管理软件开发过程的基本原则。在第六章和第十章中详细介绍了软件开发的组织和控制以及附带的一些管理和教育问题。第四章讲解了规格说明及可行性，用户要求说明，功能说明和概要系统设计的有关问题。第五章讨论了软件开发的人力和时标估算。在第七、八、九三章分别对软件开发的三个重要环节：设计、实现和质量控制进行了详细的讨论，并对已有的各种方法进行了描述和评论。

由于软件工程过程是一个涉及到顾客、具体应用领域的专家、软件工程师和管理人员等的复杂活动，在很大程度上仍然是人们的直接智力活动，依赖于信息在各方面人员之间传递的质量，因此本书在有关章节中花了相当的篇幅讨论了管理问题。由于篇幅有限，也由于人们已对设计和实现阶段提出了许多方法，因而无法一一细致介绍，对于某种方法感兴趣的读者可从附录中提到的参考书目中找到具体的介绍。另外，本书还对软件工程教育问题、软件市场、合同签订和选聘软件人员的方法进行了描述，这对于处于改革开放中的我国国情，有一定的指导意义。

本书的体系是参照A. Macro和J. Buxton的'The Craft of Software Engineering'的结构建立的，有些章节的基本内容也是从该书移植过来的。

本书的编著者为（以姓氏笔划为序）：王立峰（辽宁大学），王虹（贵州大学），方宇德（西北师范大学），刘宝祥（四平师范学院），李业丽（四平师范学院），张文波（四平师范学院），张仲选（西北大学）和杨慧（西北师范大学）。

最后全书是王立峰、刘宝祥统稿主编。辽宁大学赖翔飞教授主审。

在编写过程中，李世清校阅了第五章、第六章，并提出了不少宝贵意见，钱晓华、姜雪慧同志给予不少帮助，谨此一并致谢。由于我们水平有限，书中错误和不妥之处在所难免，敬请读者惠予指正。

编　者　著

一九八九年五月

目　　录

第一章 简 介

§1.1 信息技术，软件工程和程序员的定义

首先，需要阐明清楚“信息技术”和“软件工程”这两个术语的含义。人们通常用非常含糊的方式来使用这两个术语，对于这两个术语的含义，不少人以为是常识，但事实上却不是。象信息技术（IT）一类的许多其它同义词在广泛使用，但却没有严格的定义。考虑到这些术语出现的场合及其涉及活动的重要性，这种情况是不能令人满意的。

近来在英国，约翰·布歇尔先生主持下的一个工作小组在其关于“IT技能缺陷”报告中给出了IT的定义。这个定义由以下几个方面构成：

- 电子系统和商用电子学
- 电信和射频工程
- 计算（软件和硬件）
- 计算服务机构
- 知识库系统和人工智能
- 电子数据处理机间的通讯
- 设计和生产制造系统（与其应用相区别）

对于IT的划分可能是很困难的，特别是当它们代表的范围相互重迭，或者只有不明确的分界线，或者同出一源时更是如此。因此，布歇尔小组关于IT的定义（包括其缺点）是很有用的。它不仅简明地指出了IT的范围，而且着重地指出了计算机（从而也就指出了软件工程）的重要作用。

那么究竟什么是“软件工程”呢？事实上，这个术语更经常用来指程序设计——甚至是仅指程序设计的编码部分——而不是指范围更广的别的范畴。造成这种现象是有其历史根源的。众所周知，通过运行一系列命令使计算机硬件解决特定的问题被称为程序设计，由此人们也称程序的编写者为程序设计员。早先，程序设计员本身就是问题的解决者（应用科学家和工程师），因此，不存在定义的问题。到后来，逐渐变为用公式解决问题的人自己可以不编程，而使用别人已写好的程序（通常是用高级语言写的），因此，在某些地方，称这项工作为“编码”。在这两个没有明确定义的术语间不可避免地产生了混淆。到1968年前后，人们开始广泛地使用“软件工程”这个术语来代表包括“程序设计”和“编码”在内的一整套活动。

今天，“编码”被理解为“程序设计”的一部分，它产生用某种计算机语言表示的指令系列，以使硬件按要求去完成指定的动作；另一方面，“程序设计”是指计算机程序的设计、编码，以及由编制者将它们作为独立的程序或按需要与别的程序部分合并成为一个整体所进行的测试。

软件工程是一个完整的活动系列，其任务是在已知的资源（如时间、人力、钱、装

备等）限制下，努力生产出高质量的软件系统（即成套或不成套的程序）。这些活动包括规格说明和可行性论证（如若需要也包括样品制作），上面提到的程序设计、质量控制和检查以及本文的提供。

早在1972年，鲍尔就给软件工程下了如下的定义：

软件工程是制订并使用完整的工程法规，经济地获得可靠的、能在真的机器上运行的软件。

虽然“完整的工程法规”包括了管理方面的考虑，却是需要明确加以强调的。同样，鲍尔的定义没有明确指出支持这项“技术”所需要采用的“工具”，要知道“工具”与“工程法规”是不同的。

在当今时代，鲍尔的定义可以扩展为：软件工程是制订并使用完整的工程法规、好的管理策略，发展合适的工具和方法并在相应的地方使用它们，目的是在已知的适当的资源设备条件下，获得具有明确意义的高质量的软件。在第二章中，将给出软件工程的一个更详细的定义，其中包括软件工程作为一个过程的具有的主要的已知特性，另外，还包括由软件工程过程导出的软件系统类型的定义。

软件工程师是具有足够的有关能力和天资的人，他们职业工作的全部或主要部分就是实施软件工程。

“有关的天资”是难以定义的，但一般认为应包括：

- 象会计一样能写会算
- 类似联想能力的智慧特征
- 抽象思维能力
- 好的记忆力
- 不辞辛苦地力求精确

在给定天资因素这个前提下，软件工程师的“能力”与他们处理的软件工程任务的大小和难易，以及研制软件的环境密切相关。所谓制造软件的环境，即它是否在所有的层次上都具有好的软件工程实践所需要的、良好的计算机系统条件和软件条件，缺乏这种良好的环境的地方，被称为脆弱的工程环境。

软件工程师的能力也涉及他们对专业知识的掌握程度，其中包括能应付最不可能发生的事情并能在必要时灵活地进行引申和修改的能力。这种意思上的能力是积累形成的，它开始于初学者，继而达到初、高级软件工程师水平，最终成为这门技术的“大师”。由于历史原因，在国外最后这个阶段的大师被大多数人称为“主程序设计员”。

软件工程师的专业经历发展及课程培训方面的细目将在§10.3和§10.4中进一步讲解。在脆弱的软件工程环境中的一个主要问题是所需知识的“强化”会被不寻常地延误和阻滞。这种情况一旦发生，“软件工程师”很可能被归为“业余程序员”那一类。

并非所有的计算机程序都是由以实施软件工程为主要工作的人写的。许多主要业务是电子学的硬件工程师常常花费约10%到40%的时间去编程，这些人可称为“间歇程序员”。

就象脆弱环境中的软件工程师一样，间歇程序员不大可能具有丰富的经验、良好的

方式和方法使所需知识的强化到可行的和有效的地步。仅仅知道怎样在一台XYZ计算机上用ABC语言编程，并在去年或前年曾用几周时间做过此事是不够的，那怕编出的程序看起来似乎能“运行”的。从软件工程的观点来看，那不是用软件工程正常的过程进行的开发，而是由业余爱好者们写的（通常，业余爱好者的职业特长是别的学科）程序而已。人们把典型的业余爱好者称为“业余程序员”这是一个拥有许多种人的称呼，其中包括某些“软件工程师”和计算机科学家，以及许多硬件（电子）工程师。

IT领域的发展速度不可避免地导致业余程序员设计的高速发展。当然，这也是造成“软件工程危机”的一个主要因素。因为，软件工程是在给定资源限制（只要限制是合理的）下生产具有相当质量的软件，但业余程序设计不可能兼顾这两方面的因素。

写计算机程序的第三类人是业余癖好者。软件工程师和业余程序员编写程序是作为其工作的一部分，而业余癖好者写程序是为了教育或娱乐。通常，业余癖好者总是在个人计算机（PC）上用BASIC语言编程。

对于管理者和别的专长不是软件工程的人们来说，这类业余癖好有其积极的一面，他们将此爱好作为其专科对周围环境发展适应的一部分。但当这种业余癖好发展成为业余程序设计时，会产生不良后果。“把工作交给×××——他知道计算机的一切”是错误的说法，特别是当工作较大而且很复杂，而×××仅仅是在他的PC机上用BASIC语言写了若干个二十几条语句的程序时更是如此。近几年来，这种癖好由于得到了旨在使计算技术大众化的政府政策和计算机制造商们的支持，而大大地发展了。在每个教室放一台课桌上放一台终端，是鼓励中学生发展一些基本技能和专业兴趣的好办法，但也正如人PC，或在每张们预料的那样，在几年时间内会产生越来越多业余设计的程序。

现今，存在着软件工程危机。在本章第3节将给出造成危机的原因概述。基于上面提到的事实，其中一些原因是显而易见的。但有一个事实还没有被认识到，或者还未被知道的人所承认，那就是，自从1950年前后，开始将计算作为大规模活动以来，就一直存在着软件工程的危机。在一些时期人们认识到它的存在，在另一些时间内，则断言危机已经被解决了。

§1.2 软件工程的简短历史

毫无疑问，对复杂的应用进行程序设计是一个极其复杂和困难的事情。一个小孩就能在PC机上用BASIC编程这个事实并不能驳倒上述结论。虽然，用自来水洗厨房地板的洗涤机的控制程序只有几千条汇编指令，但却花费了人们一年半的时间去开发它。SDI“星球大战”计划要求一个多达1 000万到1亿条源代码语句的软件系统并且要求一开始就能正确运行（至少能大致正确地运行）除非能在一个多余的行星上测试。而且星球大战程序也不会用BASIC来写。

是什么东西使程序设计难以进行呢？《计算词典》（Glaser等人著于1983年）将计算机及其程序定义为：

计算机是能按清楚明白定义的方法执行一系列操作的设备。这种操作通常是数值运算或数据控制，也包括输入/输出；系列中的操作可能与数据的值有关。这样的系列即

称为程序。

按照这个定义，计算机和程序设计可以追溯到较远的时代。例如，17世纪的Leibnitz；19世纪的Jacquard，Babbage和Hollerith；20世纪中叶的Turing和Von Neumann，都对计算机作出了相当的贡献。

在1939年到1945年期间，与机械计算机不同的结合了电子部件的计算机，在英国被用在破译密码上，而在美国则主要用于计算枪炮的射击图表。大约在1950年就可以在若干家厂商那儿买到计算机，从此开始了计算机广泛使用(实际上是程序设计)的时代。之后的35年中，计算机及其使用取得了许多次重大的进展。从程序设计和程序员的观点来看，可将这个短短的历史按某些重大进展划分为4个阶段（并非是硬件"时代"的同义词）：

第一阶段（大约1950年～1958年）

在这个阶段中，首批业余程序员既是用户又是程序员，甚至还充当硬件工程师，以解决应用科学和工程领域内的问题。他们使用的是庞大而又昂贵的"主机"，并用数字码或简单的助记汇编码编程并运行。这类程序设计常常被物理学家，化学家，数学家和许多工程师看成是一种很好的智力游戏（确实也是很有趣的）。而且，这些最初的"业余程序员们"编制出许多非常聪明的、只能供编制者本人使用的程序，到如今，这些程序不是被丢弃就是被闲置在一旁（等待着编程者本人再去使用）。

第二阶段（大约1959年～1967年）

在此期间，高级语言出现并被广泛地使用，操作系统亦逐步形成，从而提高了设备的使用效率。脱机输入/输出结构的计算机使处理器从速度极慢，耗时又多的I/O操作中解放了出来。其结果使得商业数据处理（缩写为DP）也加入应用科学和工程计算数据处理的行列。这个时期内使用计算机的职业人员结构可由下述人员组成：操作员、编码员、设计员和系统分析员。用户，即第一阶段的业余程序员和DP领域的新用户，在不同程度上逐渐变得与计算机和程序无关。在此期间，发展FORTRAN，ALGOL，COBOL等高级语言的目的就是使用户的程序设计工作较容易完成。有了高级语言，用户可以不必再去编写如下的二进制编码：

```
0   000000001010   101   00000000010101000011101
```

或　　+21，5，5405

这样的十进制数码，

或　　ADD B，5

这样的记忆汇编码，在高级语言中可以写成人们更习惯的形式，

如：　　D=(A+B(I) C)

上述式子会被操作系统软件中的编译程序翻译成类似如下所示的代码：

```
CLA A
ADD B, 5
MPY C
STO D
```

这样的汇编码将被进一步译成二进制机器码。实际在机器上运行的是已被译成机器码的程序，亦即是机器码程序按事先规定的方式控制硬件电路系统动作。编码员已越来越倾

向于使用高级语言。面临要学会几种计算机语言和日益复杂的操作系统的用户，逐渐变得只是详细说明需要解决的问题，而不再亲自去充当程序员和计算机操作员了。在此期间，提出了“虚拟机”概念，当时的虚拟机看上去似乎是由操作系统软件的过程及其程序设计语言定义的。这些软件设施的发展渐渐使得硬件被日趋庞大和复杂的软件系统所包围。

在这段时间内，IBM 公司提出了“360”系列计算机（即在软件一级兼容，只是体积，价格不同的各型机器）并使其风行一时。360 系列机有分时功能，允许同时有多个用户进行操作，并拥有一整套虚拟机操作系统特征，还备有COBOL，FORTRAN，PL/I等好几种高级语言。360在商业上是一个巨大的成功，但它的操作系统软件（即 OS 360）却是一个大杂烩。OS360是大约5 000～6 000人花费了几年时间开发出来的，而且在开发期间，每年耗资5000万美元。

终于有一天，人们第一次承认了程序设计危机的存在。1968年，由北大西洋公约组织（NATO）主持召开了“软件工程”方面的第一届会议。软件工程世界会议的论文集是很值得读一读的，他们提出的问题及作出的答案对现在来说仍是非常有价值的。

第三阶段（大约1968年～1978年）

在这个时期出现了小型机并得到了广泛的使用。它的出现使计算机第一次被带到了要处理问题的面前。计算机在“传统”的 DP 领域和应用科学等方面应用的高速发展，作为电子部件的小型机的广泛使用等，使得试着从这种种现象中了解并找出软件工程方法的人们（如NATO 会议和从事计算机科学的人们）迷失方向。在此阶段中，嵌套系统的使用使业余程序设计再度活跃。进行业余程序设计的不再只是几千名应用科学家和工程师，而是数万名电子“系统”工程师。

70年代初期的业余程序员不仅对软件中需要的与大而昂贵的主机相比小且便宜的设备感兴趣，而且也对新的小型机上虚拟机软件设施的缺乏感兴趣。然而，到了 1975～1976年，形势得到了改进，小型计算机本身就拥有了较好的操作系统软件和高级语言（包括FORTRAN，ALGOL，COBOL等）。

第四阶段（大约1978年至今）

这是微处理机和微型计算机的时代，在此期间内，出现了大量的微机并得到了广泛的使用。事实上，这是使第三时期向前发展成为嵌入系统（实际上，更一般的称为IT）的一个强大动力，也是使业余程序员设计成长壮大的强有力的因素。

与小型机相比，微机的大部分硬件曾经是、至今仍是“裸露的”，即没有被多重操作软件所包围。对一些实时应用而言，微机的价值是不可非议的。但是，微机的广泛使用导致了在非常原始的开发系统（“基核”）上，或在非常裸露的目标设备上进行“肮脏的”业余程序设计。由于微机体积小，价格便宜及其潜在能力，使第三阶段产生的数万业余程序员急增到数十万。

主机和微机至今都仍在生产和使用，较大的机器主要用于具有很大的档案式存贮和检索能力的多道并发访问系统。逐渐地，较大的机器自身也结合进了微电路，这就使小型主机和大型的小型机之间的界限变得日渐模糊，就象小的小型机与大的微机一样难以区分。

然而，大型机和小型机仍然倾向具有虚拟机的配置，用于大规模的 DP 和面向计算的应用，并作为诸如嵌入式微机系统的其它应用的程序开发（宿主）环境。这些程序开发系统（通常被称为程序设计支持环境简记为 PSE），常常是非常昂贵的，一般中、小规模IT公司是无法承受的，从而使软件工程师和业余程序员都只得在目标设备自身上进行软件开发——这就是通常说的，缺乏软件工具的情形。§8.3中将进一步讨论 PSE 这方面的问题。

在此期间，高级语言的发展也没能与微机的IT应用发展保持一致。到目前为止，还没有一种能真正令人满意地写出包含完全实时的，并发等特征的软件系统的高级语言（C 语言在这些方面算是最好的，但它又很难称得上是“高级的”）。

事实上，在这段时间的一开始，E.W.Dijkstra 教授——一位杰出的计算机科学家——就提醒人们注意业余程序设计员使用便宜的小设备和原始软件开发工具研制复杂的系统可能产生的影响。其结果，他认为这会使软件工程倒退回25年，即返回到我们所说的第一阶段中去。过去的岁月证实了他的观点，重新出现了大量业余程序员。但是，这一回不再象第一阶段那样，“用户”自己编程解决他们的应用科学或机械工程计算问题，今天的业余程序员的工作成果也许会是大量销售的软件产品中的主要部分，他们一旦失败将导致严重的后果。

§1.3 软件工程当前的问题

前面是关于软件工程危机是怎样形成的一个概要性说明，下面列出软件工程当前的 6 个主要问题：

1．对于管理员和一些其它人员来说，如果他们并不熟悉软件工程但却要对之施加影响时，其能力和自信心往往是不够的。这个问题常常以所谓“代的鸿沟”这个含义被提及，它是指那些没能与课桌上的终端和家里的PC机同步成长的人们。

2．软件工程师的素质和数量都不足。在一些IT应用领域(尤其是嵌入实时系统)和某些级别的实践人员（特别是主程序设计员级）中，这种欠缺尤为突出。

3．面对高速增长的开发各种软件系统的需要，以及第 2 个问题中提及的难以保持和增加软件工程师的困难，许多管理员——尤其是那些自己缺乏能力和自信的管理人员——允许在软件开发过程中采用业余程序设计。

4．诸如PSE 和配置管理系统的软件工具仍还处于发展阶段。现在已有的软件工具或者在使用上受到限制（例如，有的PSE 就没有提供等 7 章中提到的软件设计图式表示法的功能），或者硬件及PSE软件价格是昂贵的。

5．软件工程方法仍处于发展之中。人们正在继续研究说明和设计“语言”，程序证明的规范方法，快速制作样品系统的“语言”等。

正如对程序评估和程序证明方面的一些“算法”方法一样，急于对这些研究工作作出不成熟的结论非但无助于软件工程，还可能在计算机科学家和专职软件工程师之间造成隔阂。

6．软件工程的教学方法和方式常常是不合适的。不论是国家的正规教育系统，还

是公司的“培训计划”，都没能培养出充分适应本学科的“代的鸿沟”人员。

本书讲述的内容旨在减少第1至3个问题，并帮助人们满足第6个问题提出的要求。在下节里，我们将给读者一些从实践得出的忠告。

§10.4中将给出本学科教育的几个等级，即初级、中级和调整级中的基本问题。图1.1刻画了第6个问题和软件工程危机之间的联系。

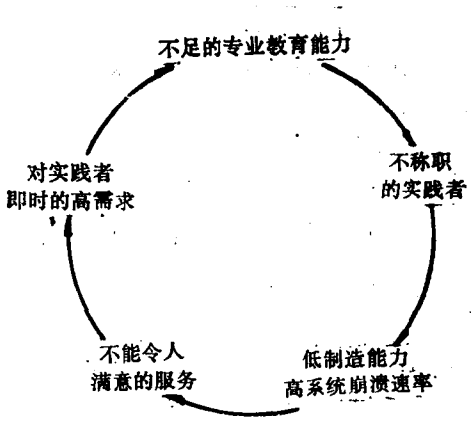

图1.1 信息技术的恶性循环

§1.4 本书采用的方法

本书并不想谈论软件工程中严格的规定和“标准”，只是想使读者对这个学科有一个相当的了解。这样做的原因有如下：

1．软件工程的应用范围多种多样，很难想象会有一种在所有环境中都非常适合的方法或规定。一个适合于小的DP应用的设计方法也许就不适用于一个大的实时系统；而适合数字运算的程序设计语言也许就不太适合文件处理应用。

2．由§1.3的第4和第5问题可知，这门学科正处于高速的变化阶段，人们正在进一步研究其方式、方法，以及研制功能又强，价格又便宜的软件工具。

我们的宗旨是多提指导方针，少讲标准。我们发现，许多人喜欢确定的方法和关于在不同环境中该怎样做的清楚明白的说明。而事实上，一本“烹调全书”不是我们的目的。我们目的旨在启发读者了解可能的最好方法。

实践工作人员，软件工程师和业余程序设计员应同样提地对待本书中提到的各个方面。也许有人轻视“管理”部分（第一，三，五，六，十章），那是因为他们对学科兴趣的偏执。这种倾向是应该克服的。一个真正的软件工程师应象能处理技术问题一样能进行管理，亦能理解别的管理员的要求。仅仅只是一个编程能手是不够的。

从各个方面影响软件开发过程的非实践者（如：管理员等），应攻读此书的全部；第四、七、八和九章的“技术”部分介绍的是必要的术语和方法，是必需熟悉的；没有比让一个人明白他所不了解的过程更糟糕的事了。我们的出发点是，假定许多不是软件工程师的人不知道在实践中怎样检查一个软件开发工作，即不懂得“结构预排”技术（这将在3.4节中讨论），我们希望通过阅读此书，上述假定不复存在。在此要强调第五章的重要性，§1.3中提到的第1个问题（即管理员，大部分工作人员和顾客缺乏专业技能）产生的一个主要影响是，经常在软件开发中对所需的时间和人力严重估计不足，而且，再没有别的什么能比这二个因素对成果的质量产生更戏剧性，更坏的影响了。

虽然第四章和第七章的例子同出一源并已被整理过了，但需要注意的是由于摘取的

部分不完整，因此并不能作为一个完善的学习实例。

在书中，我们将推荐一些好的软件工程实践的指导法则。一些进行IT软件生产的组织已经有了开发系统和软件的好的方法；但多数都还没有。总之，从1955年至今，计算机的处理能力（以每个部件在1秒钟内可执行的指令条数来衡量）已提高了100万倍，而程序员的生产能力（以单位时间完成的源代码语句数来衡量）却只增长了3倍。

有证据表明，具有相当经验和教育的实践者在有一定软件开发工具环境中，在好的软件工程指导法规的指导下，其生产高质量的软件（不仅仅是提供源代码）的能力会得到实质性的提高。我们发现经受过即使不很理想的软件工程硕士研究生班教育的人（如§10.4所述），其生产力比以前规定的能增加2～3倍。

第二章 软件工程

§2.1 软件工程的扩展定义及其说明

在§1.1中，我们给出了软件工程的一个简要的、一般的定义，并给出了编码、程序设计和软件工程之间的区别。其中软件工程是为了获得好的"软件系统"所进行的一系列活动的概括性的术语。本章旨在扩展原有的定义，并描述将软件工程视为一个处理过程时，它所具有的一些特性。

从鲍尔的一般定义开始，更深入一点讲，软件工程的含义是：

1．制造和维护软件系统。一般认为，所涉及的活动包括：规格说明；软件系统定义；软件设计，编码和程序测试；系统的整体测试和检验/证实/确认；软件维护和修改形成新版本。

这个定义包括了提供文本和测试策略的基本问题。

2．生产和维护软件系统的设计工具和方法，以及如何恰当地使用这些工具和方法。例如：关于规格说明的专门方法和处理方式；描述设计意图的符号和语言；帮助编码和测试的软件工具和实用程序，如调试工具、编辑程序等；进行文本和版本控制的结构管理系统。

3．管理整个过程，使任务能在规定的时间、经费等约束条件下经济地完成。为了使软件工程过程是可管理的，它必须由一套可见的能加以辨别的活动组成。

为了做到这一点，管理员和参与软件工程的软件工程师必须称职，这包括双方对软件工程有足够的了解，并且应能相互交流看法。

§2.2 软件工程过程及其复杂性

大多数人都认为软件工程是特殊的，并且通常也是非常困难的一种工程。但也常听到这样的问题：它真是一种工程吗？它不是一门艺术或一门科学吗？

我们偏向于把软件工程看成是一门能学会的艺术这样一个观点。一般来讲，它与工程有着很密切的联系，但是，设计阶段直观的过程使它接近艺术形式，而编码和测试阶段它又非常类似于数学的形式逻辑分支。

作为前述的第二阶段末期发生的明显的危机的结果，使人们非常注意软件工程过程，目的是了解它、改进它和它的产物——软件系统。1971年，魏伯格（Weinberg）发表了一部关于计算机程序设计心理学方面问题的著作；1974年，在魏伯格和舒尔曼（Schulman）联合发表的论文中，概要地给出了围绕软件工程所做实验的结果。下面列出的软件工程的特性，包括了他们的成果。