21世纪高等学校计算机规划教材

21st Century University Planned Textbooks of Computer Science

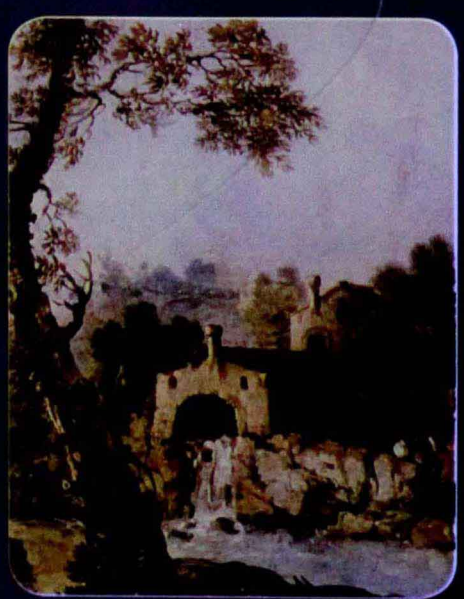
软件工程

（第4版）

Software Engineering (4th Edition)

张海藩 吕云翔 编著

- 国内软件工程领域的经典著作
- 全面介绍软件工程概念与原理
- 以丰富实例讲述软件工程方法学



名家系列

人民邮电出版社
POSTS & TELECOM PRESS

21世纪高等学校计算机规划教材

21st Century University Planned Textbooks of Computer Science

软件工程

（第4版）

Software Engineering (4th Edition)

张海藩 吕云翔 编著

名家系列

人民邮电出版社

北京

图书在版编目（CIP）数据

软件工程 / 张海藩，吕云翔编著. -- 4版. -- 北京
：人民邮电出版社，2013.9
21世纪高等学校计算机规划教材
ISBN 978-7-115-32653-9

Ⅰ. ①软… Ⅱ. ①张… ②吕… Ⅲ. ①软件工程－高等学校－教材 Ⅳ. ①TP311.5

中国版本图书馆CIP数据核字(2013)第174972号

内 容 提 要

本书是软件工程领域的经典教材。全书由5篇（16章）构成，第1篇（第1、2章）讲述软件工程与软件过程；第2篇讲述传统方法学（第3～5章），包括结构化分析、设计与实现；第3篇讲述面向对象方法学（第6～10章），包括面向对象的概念、模型、分析、设计、实现，同时介绍了统一建模语言UML；第4篇讲述软件项目管理（第11～14章），包括软件项目的计划、组织和控制，软件维护与软件文档；第5篇讲述软件工程的高级课题（第15、16章），包括形式化方法和软件重用。

本书内容新颖、实例丰富，可以作为高等院校“软件工程”课程的教材或教学参考书，也可以供程序员、软件测试工程师、系统工程师以及软件项目经理等相关人员阅读参考。

◆ 编　　著　张海藩　吕云翔
责任编辑　武恩玉
责任印制　彭志环　焦志炜

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街14号
邮编　100061　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
三河市海波印务有限公司印刷

◆ 开本：787×1092　1/16
印张：21.75　　2013年9月第4版
字数：568千字　　2013年9月河北第1次印刷

定价：45.00元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154
广告经营许可证：京崇工商广字第0021号

出版者的话

计算机科学与技术日新月异的发展，对我国高校计算机人才的培养提出了更高的要求。许多高校主动研究和调整学科内部结构、人才培养目标，提高学科水平和教学质量，精炼教学内容，拓宽专业基础，优化课程结构，改进教学方法，逐步形成了“基础课程精深，专业课程宽新”的良性格局。作为大学计算机教材建设的生力军，人民邮电出版社始终坚持服务高校教学、致力教育资源建设的出版理念，在总结前期教材建设的成功经验的同时，深入调研和分析课程体系，并充分结合我国高校计算机教育现状和改革成果，推出“推介名师好书，共享教育资源”的教材建设项目，出版了“21世纪高等学校计算机规划教材”。

本套教材的突出特点如下。

（1）作者权威 本套教材的作者均为国内计算机学科中的学术泰斗或高校教学一线的教学名师，他们有着深厚的科研功底和丰富的教学经验。可以说，这套教材汇聚了众师之精华，充分显示了这套教材的格调和品位。无论是刚入杏坛的年轻教师，还是象牙塔内的莘莘学子，细细品读其中的章节文字，定会收益匪浅。

（2）定位准确 本套教材是为普通高等院校的学生量身定做的精品教材。具体体现在：一是本套教材的作者长期从事一线科研和教学工作，对高校教学有着深刻而独到的见解；二是本套教材在选题策划阶段便多次召开调研会，对普通高校的教学需求和教材建设情况进行充分摸底，从而保证教材在内容组织和结构安排更加贴近实际教学；三是组织有关作者到较为典型的普通高等院校讲授课程教学方法，深入了解教师的教学需求，充分把握学生的理解能力，以教材内容引导授课教师严格按照科学方法实施教学。

（3）教材内容与时俱进 本套教材在充分吸收国内外最新计算机教学理念和教育体系的同时，更加注重基础理论、基本知识和基本技能的培养，集思想性、科学性、启发性、先进性和适应性于一身。

（4）一纲多本，合理配套 根据不同的教学方法，同一门课程可以有多本不同的教材，教材内容各具特色，实现教材系列资源配套。

总之，本套教材中的每一本精品教材都切实体现了各位教学名师的教学水平，充分折射出名师的教学思想，淋漓尽致地表达着名师的教学风格。我们相信，这套教材的出版发行一定能够启发年轻教师们真正领悟教学精髓，教会学生科学地掌握计算机专业的基本理论和知识，并通过实践深化对理论的理解，学以致用。

我们相信，这套教材的策划和出版，无论在形式上还是在内容上都能够显著地提高我国高校计算机专业教材的整体水平，为培养符合时代发展要求的具有较强国际竞争力的高素质创新型计算机人才，为我国的普通高等教育的计算机教材建设工作做出新的贡献。欢迎各位老师和读者给我们的工作提出宝贵意见。

第 4 版前言

20 世纪 60 年代，为了解决当时出现的“软件危机”，人们提出了软件工程的概念，并将其定义为“为了经济地获得可靠的和能在实际机器上高效运行的软件，而建立和使用的健全的工程规则”。随着 40 多年的发展，人们对软件工程逐渐有了更全面、更科学的认识，软件工程已经成为一门包括理论、方法、过程等内容的独立学科，并出现了相应的软件工程支撑工具。

然而即使在 21 世纪的今天，软件危机的种种表现依然没有彻底地得到解决，实现中很多项目依然挣扎在无法完成或无法按照规定的时间、成本，完成预期的质量的泥潭中，面临着失败的危险。究其原因，依然是软件工程的思想和方法并未深入到计算机科学技术、特别是软件开发领域中，并指导人们的开发行为。

为了振兴中国的计算机和软件产业，培养具备软件工程思想和技术，并具有相应开发经验的人才，国家近年来一直十分重视软件工程相关课程的建设和人才培养。除了开设专门的软件工程专业，也倡导在计算机科学技术相关专业开设软件工程课程，使得软件工程思想和技术在中国的 IT 人才中得到普及。

本书讲述了软件工程与软件过程，涉及传统方法学、面向对象方法学，以及软件项目管理，并且讲述了软件工程高级课题，如形式化方法（包括 Petri 网等）和软件复用。

本书第 4 版在保持原书结构和篇幅基本不变的前提下，将第 14 章“国际标准”改为“软件维护与软件文档”；每一章后的习题，也按照当前教学的需要，进行了全面的更新。

在与本书配套的教材《软件工程（第 4 版）辅导与习题解析》中，有对本书每章末的习题解析，有对软件工程各种类型的应用题的详解，以及软件工程课程设计指导，以帮助读者更好地理解和巩固所学的知识。

本书的教学安排建议如下。

章　节	内　容	学 时 数
第 1 章	软件工程概述	2
第 2 章	软件过程	2
第 3 章	结构化分析	4
第 4 章	结构化设计	4
第 5 章	结构化实现	4
第 6 章	面向对象方法学导论	4/6
第 7 章	面向对象分析	4
第 8 章	面向对象设计	4/6
第 9 章	面向对象实现	4/6
第 10 章	统一建模语言	4/6

续表

章　节	内　容	学 时 数
第11章	计划	2
第12章	组织	2
第13章	控制	2
第14章	软件维护与软件文档	2
第15章*	形式化方法	0/4
第16章*	软件重用	0/4

建议先修课程：计算机导论、面向对象程序设计、数据结构、数据库原理等。

建议理论教学时数：48～64学时。

建议实践教学时数：32～48学时。

教师可以根据教学需要适当地删除一些章节，也可根据教学目标，灵活地调整章节的顺序，增减各章的学时数。

本书作者一直在北京信息科技大学和北京航空航天大学软件学院担任软件工程课程的教学工作，进行了大量的教学探索和研究。在成书过程中，大量借鉴了笔者和同事在教学中的相关经验。在此感谢他们为此做出的贡献，也感谢其在成书过程中提供的各种宝贵资料和建议。

由于软件工程作为工程学科正处在发展与变化之中，我们力求使本书做到完美；但由于编者学习能力和水平有限，书中难免有疏漏之处，恳请各位同仁和广大读者给予批评指正，也希望各位能将使用此教材过程中的经验和心得与我们交流（yunxianglu@hotmail.com）。

编　者

2013年6月

目 录

第 1 篇 软件工程与软件过程

第 2 篇 传统方法学

第3篇 面向对象方法学

第 1 篇　软件工程与软件过程

第 1 章　软件工程概述

人类社会已经跨入了 21 世纪，计算机系统已经渗入人类生活的各个领域，同时计算机软件已经发展成为当今世界最重要的技术领域。研究软件本身则产生了一门重要的学科就是软件工程。软件工程的研究领域包括软件的开发方法、软件的生命周期以及软件的工程实践等。

1.1　软件危机与软件工程的起源

1.1.1　计算机系统的发展历程

20 世纪 60 年代中期以前，是计算机系统发展的早期。在这个时期通用硬件已经相当普遍，软件却是为每个具体应用而专门编写的，大多数人认为软件开发是无须预先计划的事情。这时的软件实际上就是规模较小的程序，程序的编写者和使用者往往是同一个（或同一组）人。由于规模小，程序编写起来相当容易，也没有什么系统化的方法，对软件开发工作更没有进行任何管理。这种个体化的软件环境，使得软件设计往往只是在人们头脑中隐含进行的一个模糊过程，除了程序清单之外，根本没有其他文档资料保存下来。

从 20 世纪 60 年代中期到 70 年代中期，是计算机系统发展的第二代。在这 10 年中计算机技术有了很大进步。多道程序、多用户系统引入了人—机交互的新概念，开创了计算机应用的新境界，使硬件和软件的配合上了一个新的层次。实时系统能够从多个信息源收集、分析和转换数据，从而使得进程控制能以毫秒而不是分钟来进行。在线存储技术的进步导致了第一代数据库管理系统的出现。

计算机系统发展的第二代的一个重要特征是出现了“软件作坊”，广泛使用产品软件。但是，“软件作坊”基本上仍然沿用早期形成的个体化软件开发方法。随着计算机应用的日益普及，软件数量急剧膨胀。在程序运行时发现的错误必须设法改正；用户有了新的需求时必须相应地修改程序；硬件或操作系统更新时，通常需要修改程序以适应新的环境。上述种种软件维护工作，以令人吃惊的比例耗费资源。更严重的是，许多程序的个体化特性使得它们最终成为不可维护的。“软件危机”就这样开始出现了。1968 年北大西洋公约组织的计算机科学家在联邦德国召开国际会议，讨论软件危机问题。在这次会议上正式提出并使用了“软件工程”这个名词，一门新兴的工程学科就此诞生了。

1.1.2 软件危机介绍

软件危机是指在计算机软件的开发和维护过程中所遇到的一系列严重问题。这些问题绝不仅仅是不能正常运行的软件才具有的。实际上，几乎所有软件都不同程度地存在这些问题。概括地说，软件危机包含下述两方面的问题：如何开发软件，以满足对软件日益增长的需求；如何维护数量不断膨胀的已有软件。鉴于软件危机的长期性和症状不明显的特征，近年来有人建议把软件危机更名为“软件萧条（depression）”或“软件困扰（affliction）”。不过“软件危机”这个词强调了问题的严重性，而且也已为绝大多数软件工作者所熟悉，所以本书仍将沿用它。

具体来说，软件危机主要有以下一些典型表现。

① 对软件开发成本和进度的估计常常很不准确。实际成本比估计成本有可能高出一个数量级，实际进度比预期进度拖延几个月甚至几年的现象并不罕见。这种现象降低了软件开发组织的信誉。而为了赶进度和节约成本所采取的一些权宜之计又往往损害了软件产品的质量，从而不可避免地会引起用户的不满。

② 用户对“已完成的”软件系统不满意的现象经常发生。软件开发人员常常在对用户要求只有模糊的了解，甚至对所要解决的问题还没有确切认识的情况下，就仓促上阵匆忙着手编写程序。软件开发人员和用户之间的信息交流往往很不充分，“闭门造车”必然导致最终的产品不符合用户的实际需要。

③ 软件产品的质量往往靠不住。软件可靠性和质量保证的确切定量概念刚刚出现不久，软件质量保证技术还没有坚持不懈地应用到软件开发的全过程中，这些都导致软件产品发生质量问题。

④ 软件常常是不可维护的。很多程序中的错误是非常难改正的，实际上不可能使这些程序适应新的硬件环境，也不能根据用户的需要在原有程序中增加一些新的功能。“可重用的软件”还是一个没有完全做到的、正在努力追求的目标，人们仍然在重复开发类似的或基本类似的软件。

⑤ 软件通常没有适当的文档资料。计算机软件不仅仅是程序，还应该有一整套文档资料。这些文档资料应该是在软件开发过程中产生出来的，而且应该是“最新式的”（即和程序代码完全一致的）。软件开发组织的管理人员可以使用这些文档资料作为里程碑（milestone），来管理和评价软件开发工程的进展状况；软件开发人员可以利用它们作为通信工具，在软件开发过程中准确地交流信息；对于软件维护人员而言，这些文档资料更是至关重要、必不可少的。缺乏必要的文档资料或者文档资料不合格，必然给软件开发和维护带来许多严重的困难和问题。

⑥ 软件成本在计算机系统总成本中所占的比例逐年上升。由于微电子学技术的进步和生产自动化程度不断提高，硬件成本逐年下降，然而软件开发需要大量人力，软件成本随着通货膨胀以及软件规模和数量的不断扩大而持续上升。美国在1985年软件成本大约已占计算机系统总成本的90%。

⑦ 软件开发生产率提高的速度，既跟不上硬件的发展速度，也远远跟不上计算机应用迅速普及深入的趋势。软件产品“供不应求”的现象，使人类不能充分利用现代计算机硬件提供的巨大潜力。

以上列举的仅仅是软件危机的一些明显的表现，与软件开发和维护有关的问题远远不止这些。

1.1.3 产生软件危机的原因

在软件开发和维护的过程中存在这么多严重问题，一方面与软件本身的特点有关，另一方面也和软件开发与维护的方法不正确有关。

软件不同于硬件，它是计算机系统中的逻辑部件而不是物理部件。由于软件缺乏“可见性”，在写出程序代码并在计算机上试运行之前，软件开发过程的进展情况较难衡量，软件的质量也较

难评价，因此，管理和控制软件开发过程相当困难。此外，软件在运行过程中不会因为使用时间过长而被“用坏”，如果运行中发现错误，很可能是遇到了一个在开发时期引入的在测试阶段没能检测出来的错误，因此，软件维护通常意味着改正或修改原来的设计，这就在客观上使得软件较难维护。

软件不同于一般程序，它的一个显著特点是规模庞大，而且程序复杂性将随着程序规模的增加而呈指数上升。为了在预定时间内开发出规模庞大的软件，必须由许多人分工合作。然而，如何保证每个人完成的工作合在一起确实能构成一个高质量的大型软件系统，更是一个极端复杂困难的问题，不仅涉及许多技术问题，诸如分析方法、设计方法、形式说明方法、版本控制等，更重要的是必须有严格而科学的管理。

软件本身独有的特点确实给开发和维护带来一些客观困难，但是人们在开发和使用计算机系统的长期实践中，也确实积累和总结出了许多成功的经验。如果坚持不懈地使用经过实践考验证明是正确的方法，许多困难是完全可以克服的，过去也确实有一些成功的范例。但是，目前相当多的软件专业人员对软件开发和维护还有不少糊涂观念，在实践过程中或多或少地采用了错误的方法和技术，这可能是使软件问题发展成软件危机的主要原因。

与软件开发和维护有关的许多错误认识和做法的形成，可以归于在计算机系统发展的早期阶段软件开发的个体化特点。错误的认识和做法主要表现为忽视软件需求分析的重要性，认为软件开发就是编写程序并设法使之运行，轻视软件维护等。

事实上，对用户要求没有完整准确的认识就匆忙着手编写程序是许多软件开发工程失败的主要原因之一。只有用户才真正了解他们自己的需要，但是许多用户在开始时并不能准确具体地叙述他们的需要，软件开发人员需要做大量深入细致的调查研究工作，反复多次地和用户交流信息，才能真正全面、准确、具体地了解用户的要求。对问题和目标的正确认识是解决任何问题的前提和出发点，软件开发同样也不例外。急于求成，仓促上阵，对用户要求没有正确认识就匆忙着手编写程序，这就如同不打好地基就盖高楼一样，最终必然垮台。事实上，越早开始编写程序，完成它所需要用的时间往往越长。

一个软件从定义、开发、使用和维护，直到最终被废弃，要经历一个漫长的时期，这就如同一个人要经过胎儿、儿童、青年、中年和老年，直到最终死亡的漫长时期一样。通常把软件经历的这个漫长的时期称为生命周期。软件开发最初的工作应是问题定义，也就是确定要求解决的问题是什么；然后要进行可行性研究，决定该问题是否存在一个可行的解决办法；接下来应该进行需求分析，也就是深入具体地了解用户的要求，在所要开发的系统（不妨称之为目标系统）必须做什么这个问题上和用户取得完全一致的看法。经过上述软件定义时期的准备工作才能进入开发时期，而在开发时期首先需要对软件进行设计（通常又分为概要设计和详细设计两个阶段），然后才能进入编写程序的阶段，程序编写完之后还必须经过大量的测试工作（需要的工作量通常占软件开发全部工作量的 40%～50%）才能最终交付使用。所以，编写程序只是软件开发过程中的一个阶段，而且在典型的软件开发工程中，编写程序所需的工作量只占软件开发全部工作量的 10%～20%。

另一方面还必须认识到程序只是完整的软件产品的一个组成部分，在上述软件生命周期的每个阶段都要得出最终产品的一个或几个组成部分（这些组成部分通常以文档资料的形式存在）。也就是说，一个软件产品必须由一个完整的配置组成，软件配置主要包括程序、文档、数据等成分。必须清除只重视程序而忽视软件配置其余成分的糊涂观念。

做好软件定义时期的工作，是降低软件成本提高软件质量的关键。如果软件开发人员在定义时期没有正确全面地理解用户需求，直到测试阶段或软件交付使用后才发现“已完成的”软件不

完全符合用户的需要，这时再修改就为时晚矣。

严重的问题是，在软件开发的不同阶段进行修改需要付出的代价是很不相同的。在早期引入变动，涉及的面较少，因而代价也比较低；在开发的中期，软件配置的许多成分已经完成，引入一个变动要对所有已完成的配置成分都做相应的修改，不仅工作量大，而且逻辑上也更复杂，因此付出的代价剧增；在软件“已经完成”时再引入变动，当然需要付出更高的代价。根据美国一些软件公司的统计资料，在后期引入一个变动比在早期引入相同变动所需付出的代价高2～3个数量级。图1.1所示为在不同时期引入同一个变动需要付出的代价随时间变化的趋势。

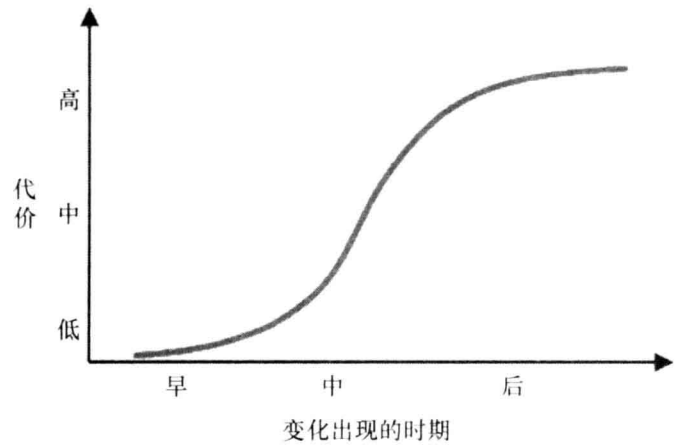


图1.1　引入同一个变动付出的代价随时间变化的趋势

通过上面的论述不难认识到，轻视维护是一个最大的错误。许多软件产品的使用寿命长达10年甚至20年，在这样漫长的时期中不仅必须改正使用过程中发现的每一个潜伏的错误，而且当环境变化时（如硬件或系统软件更新换代）还必须相应地修改软件以适应新的环境，特别是必须经常改进或扩充原来的软件以满足用户不断变化的需要。所有这些改动都属于维护工作，而且是在软件已经完成之后进行的，因此，维护是极端艰巨复杂的工作，需要花费很大代价。统计数据表明，实际上用于软件维护的费用占软件总费用的55%～70%。软件工程学的一个重要目标就是提高软件的可维护性，减少软件维护的代价。

了解产生软件危机的原因，澄清错误认识，建立起关于软件开发和维护的正确概念，还仅仅是解决软件危机的开始，全面解决软件危机需要一系列综合措施。

1.1.4　消除软件危机的途径

为了消除软件危机，首先应该对计算机软件有一个正确的认识。正如1.1.3小节中讲过的，应该彻底清除在计算机系统早期发展阶段形成的“软件就是程序”的错误观念。一个软件必须由一个完整的配置组成。事实上，软件是程序、数据及相关文档的完整集合。其中，程序是能够完成预定功能和性能的可执行的指令序列；数据是使程序能够适当地处理信息的数据结构；文档是开发、使用和维护程序所需要的图文资料。1983年IEEE（电气和电子工程师协会）为软件下的定义是：计算机程序、方法、规则、相关的文档资料以及在计算机上运行程序时所必需的数据。虽然表面上看来在这个定义中列出了软件的5个配置成分，但是，方法和规则通常是在文档中说明并在程序中实现的。

更重要的是，必须充分认识到软件开发不是某种个体劳动的神秘技巧，而应该是一种组织良好、管理严密、各类人员协同配合、共同完成的工程项目。必须充分吸取和借鉴人类长期以来从事各种工程项目所积累的行之有效的原理、概念、技术和方法，特别要吸取几十年来人类从事计

算机硬件研究和开发的经验教训。

应该推广和使用在实践中总结出来的开发软件成功的技术和方法，并且研究探索更好、更有效的技术和方法，尽快消除在计算机系统早期发展阶段形成的一些错误概念和做法。

应该开发和使用更好的软件工具。正如机械工具可以“放大”人类的体力一样，软件工具可以“放大”人类的智力。在软件开发的每个阶段都有许多烦琐重复的工作需要做，在适当的软件工具辅助下，开发人员可以把这类工作做得既快又好。如果把各个阶段使用的软件工具有机地集合成一个整体，支持软件开发的全过程，则称为软件工程支撑环境。

总之，为了消除软件危机，既要有技术措施（方法和工具），又要有必要的组织管理措施。软件工程正是从管理和技术两方面研究如何更好地开发和维护计算机软件的一门新兴学科。

1.2　软 件 工 程

1.2.1　什么是软件工程

概括地说，软件工程是指导计算机软件开发和维护的工程学科。采用工程的概念、原理、技术和方法来开发与维护软件，把经过时间考验而证明正确的管理技术和当前能够得到的最好的技术方法结合起来，经济地开发出高质量的软件并有效地维护它，这就是软件工程。

下面给出软件工程的几个定义。

1983 年 IEEE 给软件工程下的定义是：“软件工程是开发、运行、维护和修复软件的系统方法。”这个定义相当概括，它主要强调软件工程是系统方法而不是某种神秘的个人技巧。

Fairly 认为：“软件工程学是为了在成本限额以内按时完成开发和修改软件产品所需要的系统生产和维护技术及管理学科。”这个定义明确指出了软件工程的目标是在成本限额内按时完成开发和修改软件的工作，同时也指出了软件工程包含技术和管理两方面的内容。

Fritz Bauer 给出了下述定义：“软件工程是为了经济地获得可靠的且能在实际机器上有效地运行的软件，而建立和使用的完善的工程化原则。”这个定义不仅指出软件工程的目标是经济地开发出高质量的软件，而且强调了软件工程是一门工程学科，它应该建立并使用完善的工程化原则。

1993 年 IEEE 进一步给出了一个更全面的定义。

软件工程是：①把系统化的、规范的、可度量的途径应用于软件开发、运行和维护的过程，也就是把工程化应用于软件中；②研究①中提到的途径。

认真研究上述这些关于软件工程的定义，有助于我们建立起对软件工程这门工程学科的全面的整体性认识。

1.2.2　软件工程的基本原理

自从 1968 年在联邦德国召开的国际会议上正式提出并使用了“软件工程”这个术语以来，研究软件工程的专家学者们陆续提出了 100 多条关于软件工程的准则或“信条”。著名的软件工程专家 Barry W. Boehm 综合这些学者们的意见并总结了 TRW 公司多年开发软件的经验，于 1983 年在一篇论文中提出了软件工程的 7 条基本原理。他认为这 7 条原理是确保软件产品质量和开发效率原理的最小集合。这 7 条原理是互相独立的，其中任意 6 条原理的组合都不能代替另一条原理，因此，它们是缺一不可的最小集合。然而这 7 条原理又是相当完备的，人们虽然不能用数学方法

严格证明它们是一个完备的集合，但是可以证明在此之前已经提出的100多条软件工程原理都可以由这7条原理的任意组合蕴含或派生。

下面简要介绍软件工程的7条基本原理。

1. 用分阶段的生命周期计划严格管理

有人经统计发现，在不成功的软件项目中有一半左右是由于计划不周造成的，可见把建立完善的计划作为第1条基本原理是吸取了前人的教训而提出来的。

在软件开发与维护的漫长生命周期中，需要完成许多性质各异的工作。这条基本原理意味着，应该把软件生命周期划分成若干个阶段，并相应地制定出切实可行的计划，然后严格按照计划对软件的开发与维护工作进行管理。Boehm认为，在软件的整个生命周期中应该制定并严格执行6类计划，它们是项目概要计划、里程碑计划、项目控制计划、产品控制计划、验证计划和运行维护计划。

不同层次的管理人员都必须严格按照计划各尽其职地管理软件开发与维护工作，绝不能受客户或上级人员的影响而擅自背离预定计划。

2. 坚持进行阶段评审

当时已经认识到，软件的质量保证工作不能等到编码阶段结束之后再进行。这样说至少有两个理由：第一，大部分错误是在编码之前造成的，如根据Boehm等人的统计，设计错误占软件错误的63%，编码错误仅占37%；第二，错误发现与改正得越晚，所需付出的代价也越高（参见图1.1）。因此，在每个阶段都进行严格的评审，以便尽早发现在软件开发过程中所犯的错误，是一条必须遵循的重要原则。

3. 实行严格的产品控制

在软件开发过程中不应随意改变需求，因为改变一项需求往往需要付出较高的代价。但是，在软件开发过程中改变需求又是难免的。由于外部环境的变化，相应地改变用户需求是一种客观需要，显然不能硬性禁止客户提出改变需求的要求，而只能依靠科学的产品控制技术来顺应这种要求。也就是说，当改变需求时，为了保持软件各个配置成分的一致性，必须实行严格的产品控制，其中主要是实行基准配置管理。所谓基准配置又称为基线配置，它们是经过阶段评审后的软件配置成分（各个阶段产生的文档或程序代码）。基准配置管理也称为变动控制：一切有关修改软件的建议，特别是涉及对基准配置的修改建议，都必须按照严格的规程进行评审，获得批准以后才能实施修改。绝对不能谁想修改软件（包括尚在开发过程中的软件），就随意进行修改。

4. 采用现代程序设计技术

从提出软件工程的概念开始，人们一直把主要精力用于研究各种新的程序设计技术。20世纪60年代末提出的结构程序设计技术，已经成为绝大多数人公认的先进的程序设计技术。以后又进一步发展出各种结构分析（structured analysis，SA）与结构设计（structured design，SD）技术。近年来，面向对象技术已经在许多领域中迅速地取代了传统的结构化开发方法。实践表明，采用先进的技术不仅可以提高软件开发和维护的效率，而且可以提高软件产品的质量。

5. 结果应能清楚地审查

软件产品不同于一般的物理产品，它是看不见摸不着的逻辑产品。软件开发人员（或开发小组）的工作进展情况可见性差，难以准确度量，从而使得软件产品的开发过程比一般产品的开发过程更难于评价和管理。为了提高软件开发过程的可见性，更好地进行管理，应该根据软件开发项目的总目标及完成期限，规定开发组织的责任和产品标准，从而使得所得到的结果能够清楚地审查。