第1章 软件工程概述

本章导读

作为一个初学者，你知道软件工程产生的背景吗？软件工程到底是干什么的？软件工程到底干些什么？本章作为开门篇，将为你解答这些疑问，逐步引领你走进软件工程的精彩世界。

（1）了解软件的概念和软件产品的特点，了解软件产品加工的特殊性。

（2）了解软件开发的发展历程，在发展中逐步暴露了软件产品开发方法和管理中的问题即软件危机。

（3）了解解决软件危机催生了软件工程。

（4）掌握软件工程的概念、原理和内容。

（5）介绍软件技术和软件工程之间的联系，了解技术进步推动着软件工程的发展。

（6）从软件工程的发展了解传统软件工程与现代软件工程。

1.1 软件的概念

1.1.1 软件和软件产品

1. 软件

有关软件的定义，著名的权威组织和学者都给过不同且相似的基本定义。1983 年，IEEE 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。软件工程的先驱者，美国 R.S. Pressman & Associates 公司总裁

Roger S. Pressman 教授，在《Software Engineering: A Practitioner's Approach》[1]一书中关于软件的定义是：软件是①能够完成预定功能和性能的可执行的指令（计算机程序）；②使得程序能够适当地操作信息的数据结构；③描述程序的操作和使用的文档。总之，现在人们已经改变了“软件＝程序”的错误认识，普遍接受“软件＝程序+数据+文档”的概念。

依现代软件的发展看，程序不仅有可运行的代码，还要有程序运行的配置环境；数据不仅指程序中应用到的数据结构，还有数据的存储；文档指的是在软件开发和维护过程中形成的文字记录，包括从软件计划、分析、设计、实现（编码与测试）和维护等活动中形成的所有文件，乃至用户使用说明书、在线帮助等，这些文档相当于物理产品在不同加工阶段提供的工件。

程序和数据是软件运行的基本要素（基因），而文档则是延续软件运行寿命的资料保障。当软件需要纠正基因缺陷使它变得更强壮、当软件需要生长进化扩充它的行为能力时都需要依靠文档，如果没有依靠文档提供的信息来做这些事情等于在加速软件的消亡。因此文档相当于软件的“基因信息”、“医疗记录”和“保健剂方”。有关各阶段具体文档的作用、规范和建立详见后续章节的内容。有关文档的管理详见 9.5 节。

2. 软件产品

软件是通过加工得到的一种逻辑产品，除了物理产品所具有的生命期、可出售、需维护等一些属性外，软件产品还具有如下一些特点。

1）软件产品自身特点

- 软件是抽象的逻辑产品而不是实物产品，没有明显的物理加工过程，它是人的高智力劳动，把人的知识和技术、环境信息高度密集在一起的产品。因此检测劳动者工作进度、工作效率、工作质量等指标的可观测性低，造成对开发过程的进展情况难以衡量、质量难以评价、开发过程的管理和控制困难、成本和工期估算困难。有的软件加工者如同制作“皇帝新衣”的裁缝，看着挺卖力气却可能什么都没做；有的在长期深思熟虑，看似没有什么成效，但可能一夜间灵感爆发，许多问题迎刃而解。
- 软件功能的实现与硬件和支撑环境密切相关，不能单独存在。
- 软件产品研制成功后，可轻易地进行大量复制，知识产权保护困难。
- 软件不会被磨损，但是随着使用时间的推移易出现不适应新的软件生态环境而相对退化。
- 在测试阶段难以发现的隐藏错误，在投入运行后不断暴露。因此补丁不断，维护越来越困难，费用越来越高。

2）业界环境特点

- 软件开发者和用户知识领域不同，造成交流上的困难。不断修改需求，致使双方厌倦，工程流产。
- 大型应用软件开发工期长于软件平台技术的更新周期，往往刚开发完就已经是落后的产品。
- 构件与复用技术并不理想，在 IT 公司内部通过开发积累可利用一些复用，但还没有做到如同物理产品那样可在市场上供应和购买标准构件。
- 开发人员流动性大，质量和工期难免受到影响。

1.1.2　软件产品的类型

生活在信息社会里，人们的工作、交流、娱乐、出行等一切活动都离不开软件产品支持，可能时时刻刻都在同形形色色、五花八门、种类繁多的软件打交道。根据不同的分类规则会得到不同的分类结果。如按工作方式分有实时软件、交互式软件、批处理式软件等；按服务对象分有商业通用、企业定制、个人办公、工程应用、科学计算、人工智能等软件。本书是从软件工程的角度划分如下三个层面：基础软件、应用软件、支撑软件。这样的分类可以用来理解不同层面的软件开发组织其责任和方法是不尽相同的，其能力成熟度也是不同的。

1. 基础软件

基础软件是保证计算机本身运行和支撑应用软件运行的基础环境。

1）系统软件

系统软件是用来管理、控制和维护计算机的各种资源，并使其充分发挥作用的基础环境，也是其他软件运行的基础软件。

- 操作系统：如 DOS、Windows、Linux、UNIX 等。
- 语言处理系统：编译程序、解释程序和汇编程序，如 MASM、JDK。

2）数据库

提供数据存储及访问的管理功能的系统，称为数据库管理系统（Database Management System，DBMS）。如 Oracle、SQL Server、DB2、Informix、MySQL 等国外数据库软件，国产的数据库软件有东软的 OpenBASE、金仓的 KingbaseES 等。

3）中间件

中间件是继操作系统和数据库管理系统之后随着网络的兴起和发展而新兴的一种基础软件。在网络环境下应用软件的运行需要并发、事务、安全、对象生命期等高层操作，应用软件的这些高层操作不直接与操作系统或数据库打交道，而是由介于操作系统（或数据库）与应用软件之间的中间层来管理，称为中间件，中间件即是管理应用软件的运行容器。如微软的 IIS、IBM 的 WebSphere、BEA 的 WebLogic，以及 Apache 的开源 Tomcat 等。本书所讨论的架构就与中间件有关，详见 11.1.3 节。

这个层面的软件开发由引领软件开发走向的世界顶级开发商提供，如 Microsoft、IBM、Sun、Oracle 等。

2. 支撑软件

支撑软件是在系统软件和应用软件之间，提供应用软件设计、开发、测试、评估、运行检测等辅助功能的软件。

这个层面的软件也都是由世界顶级软件供应商开发的。

1）编程工具

如微软的 Visual Studio；Borland 的 C++ Builder、JBuilder；IBM 提供的开源集成开发环境 Eclipse 都受到广大开发者的普遍欢迎。

2）建模工具

如 IBM Rational Rose 的软件建模；KBSI 的 IDEF 企业建模；Sybase 的 PowerDesigner

数据建模；IDS Scheer 的流程建模 ARIS[4]；Microsoft Office Visio 也提供 UML、IDEF 等建模工具集。

3）版本控制

如 IBM Rational 套装中的 ClearCase；Borland StarTeam；UNIX、Linux、Windows 不同版本的 CVS；基于 Apache 管理平台的开源 SVN[5]。

4）软件测试

如 IBM 公司 Rational 系列测试工具 ClearQuest、Robot 等；Mercury 公司的 WinRunner、QuickTest Pro、LoadRunner；Segue 公司的 TestDirector。

3. 应用软件

应用软件是将客观世界中的物理存在或逻辑存在（人类的组织和活动）表达在计算机世界里满足特定需求并且能够运行和维护的载体集合。

这些载体就是程序+数据+文档，也就是通常定义下的软件。事实上软件工程所涉及的就是这个层面的软件开发方法学和管理学。对于一般的 IT 企业主要也就是开发这个层面的应用软件。这个层面又大致分为以下两类。

1）桌面通用软件（个人软件）

安装在 PC 上可单独运行的软件，如 WPS 和微软的 Office 办公软件，还有 Adobe、AutoCAD、Photoshop、Nero、多媒体、单机版游戏等类似的通用软件。这类软件的特点是单人操作、通用性强，但算法较为复杂。

2）企业级软件

企业级应用是指那些为商业组织、大型企业、特定业务领域、行政管理、事业部门而创建并部署的软件。企业级应用软件源于早期的“信息系统”、“数据处理”。早期的企业软件开发集中于部门级的，如“工资管理系统”、“材料管理系统”、“设备管理系统”等。随着经济全球化和信息时代的到来，企业级应用软件已经从部门独立应用发展到企业集成，从企业内部发展到企业间，从少数人使用发展到面向广大客户。诸如企业资源规划 ERP、客户关系管理 CRM、供应链管理 SCM 等，还有金融、电信、政府 OA、电子商务等各种专用软件。这些大型企业级应用的软件功能复杂，涉及的外部资源众多、事务密集、数据量大、用户数多、并发操作，从而构成一个结构复杂、操作界面友好、跨越 Intranet 和 Internet 的分布式企业应用集群，有较强的安全性考虑。由此也造就了企业级软件在开发上具有共同的特点：软件为分布式多层结构。本书所要奉献给读者的就是针对这样的企业级软件开发如何应用架构技术处理好这些层的分析和设计。本书重点介绍的软件开发就是针对企业级软件，参见 11.1 节。

4. 嵌入式软件

手机、PDA、数码相机、GPS 等智能家用电器这样的嵌入式设备在人们的日常生活中随处可见，在航空航天、汽车装配、钢铁冶金、石油化工等高科技生产领域中的基础自动化以及医学检测、媒体通信等装备上更是离不开嵌入式设备。随着嵌入式技术及应用市场的迅速发展，嵌入式软件根据自身特点已形成独立的发展分支。目前有单独的“嵌入式软件工程”。嵌入式也有自己的基础软件（嵌入式操作系统，如适用于移动设备的 Android 系统）、支撑软件（开发器提供编辑、编译/反编译、上传/下载、API 调用等），以及针对不同

设备的应用软件。这类软件需要较高的可靠型、实时性、精确性，算法较为复杂。

1.2　软件的发展与软件危机的产生

本节通过介绍软件的发展过程了解软件危机的产生原因及软件工程诞生的背景。

1.2.1 软件的发展过程

参照文献[1]以及一些国内外教科书，将软件从 20 世纪 50 年代开始的 50 年发展历程划分为 4 个阶段，如图 1-1 所示。当然这种阶段的年代划分指的是国外，在早期的三个阶段，我国经历的每个阶段相对国外要延迟一个年代。尽管我国的软件行业起步晚并且现在仍然落后，但自 20 世纪 80 年代改革开放以来，由于信息传播的便利及商业利益驱动，延迟的差距正在缩小。在 21 世纪我国某些软件企业已开始同国外顶级软件开发商共同合作开展项目，某些方面已经同国外保持同步发展的态势。

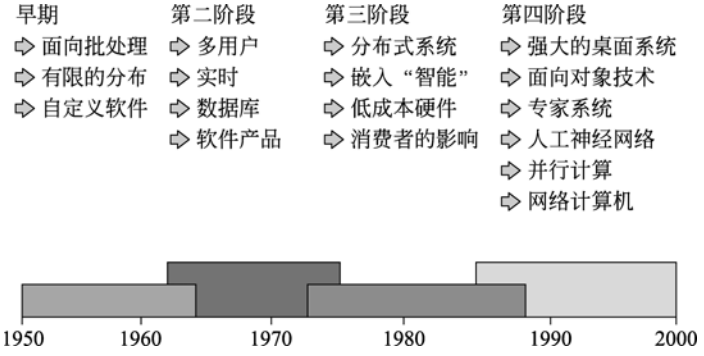


图 1-1 软件发展的 4 个阶段

为了简要说明软件危机的产生和软件工程的诞生原因，这里以表格形式简单介绍前三个阶段的软件发展过程，如表 1-1 所示。

表 1-1 软件开发早期的三个阶段

阶 段	加 工 特 点	软 件 特 点	主 要 用 途
第一阶段 1950 — 1960 （个体式）	• 个体化编程，依靠个人想法和智慧。 • 只有程序设计概念，没有过程和方法。 • 只有源代码、无其他文档。 • 语言低级，工具简单	小规模； 特定目的专用	科学计算，不追求商业效应
第二阶段 1960 — 1970 （作坊式）	• 小组合作式“软件作坊”，基本上仍然沿用早期的个体化软件开发方式。 • 有了软件设计的概念，采用结构化方法，但不讲究过程，没形成软件开发的理论体系。 • 出现高级语言和数据库技术	规模变大、结构复杂、成本增高、维护困难；导致烂尾工程、项目流产，失败的项目越来越多，软件危机逐渐显现	应用于企业； 可以作为商品出售，可以赚钱
第三阶段 始于 20 世纪 70 年代中期至 80 年代后期（团队式）	• 形成规模化团队开发，出现软件业。 • 软件开发技术有很大进步，面向对象开发技术出现，开发工具、网络环境、分布式计算有很大发展。 • 未能获得突破性进展，软件开发技术的进步发展一直未能满足日益增长的软件需求	软件的数量及规模急剧膨胀；软件功能及运行环境复杂；开发难度越来越大，找不到解决维护的办法，失败的软件开发项目屡见不鲜；软件危机的表现突出	面向市场的软件商品已形成一定的市场规模；主要市场需求是企业合同式定制软件，解决企业的信息管理

1.2.2　软件危机

软件开发早期发展的前三个阶段，逐步暴露出软件危机的现象，如表 1-1 所示。

1. 软件危机的概念

软件危机指的是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。

“软件危机”的原文是：Software Crisis。这个名词是 1968 年在德国召开的 NATO（North Atlantic Treaty Organization，北大西洋公约组织）会议上由计算机科学家首次提出的。这一概念的提出，使得人们开始对软件及其特性进行研究，改变了对软件的不正确看法，认识到软件开发需要工程化管理。同时，讨论和制定了摆脱“软件危机”的对策，会议上第一次提出了软件工程（Software Engineering）这个概念，希望用工程化的原则和方法来克服软件危机，从此一门新兴的工程学科——“软件工程学”应运而生。

由于美国当时缺乏软件人员，计算机公司大量招聘程序员，甚至公共汽车司机也被招去，因此粗制滥造的软件大量涌向市场，致使软件出现大量错误，质量难以保证，进度很难控制，软件开发周期延长，维护成本剧增，这就是软件危机。1968 年，有人在一次计算机软件学术会上说：“整个事业是建立在一个大骗局上”。可见软件危机已发展到何种程度，它已明显地影响到社会的发展。计算机科学在软件危机中挣扎，社会在为软件危机付出沉重的代价[6]。

最为突出广为流传的案例是美国 IBM 公司于 1963—1966 年开发的 IBM360 系列机的操作系统。该项目的负责人 Fred Brooks 在总结该项目时无比沉痛地说：“……正像一只逃亡的野兽落到泥潭中做垂死挣扎，越是挣扎，陷得越深，最后无法逃脱灭顶的灾难……程序设计工作正像这样一个泥潭……一批批程序员被迫在泥潭中拼命挣扎……谁也没有料到问题竟会陷入这样的困境……”IBM360 操作系统的历史教训已成为软件开发项目中的典型事例被记入历史史册[7]。

大量软件开发项目的失败和经济损失，使软件专家明确认识到软件开发必须以新的方法作为指导，原有的软件开发技术手段和管理方法必须改变，从此孕育着软件工程时代的诞生。

软件危机包含两方面问题：第一是软件开发方面的问题，软件规模日趋增大、内容日趋复杂，开发困难；第二是软件维护方面的问题，需求和环境不断变化，使得维护数量不断膨胀；在落后的软件生产方式下，出现了软件开发时间和成本难以控制、维护困难、成本攀升等一系列严重问题。

解决软件危机就是解决这两方面问题：①如何开发软件，以满足不断增长、日趋复杂的需求；②如何维护数量不断膨胀的软件产品。

2. 软件危机产生的原因

从哲学上看，一切矛盾均来自于主客观不相适应。软件危机的产生也是由于主观上的观念和开发、维护方法不适应客观对软件需求的发展。

1）客观方面

- 软件是逻辑产品，与物理产品加工有很大区别。
- 计算机发展迅速，软件需求规模庞大、数量增加、功能复杂。

- 业务流程不断改造，需求不断变化，维护量增大。

2）主观方面

传统观念还没有被突破。还没认识到软件生产需要管理，还没有认识到逻辑产品加工的内在规律和找到恰当的规则，还凝滞在习惯的物理加工思维及管理模式上而没有适应到逻辑产品的生产中。因而造成管理上的困难会有如下表现。

- 脱离工程化管理。
- 管理理念和方法的落后。
- 开发人员逻辑思维的跟进不及时。
- 延续使用个体化开发方法。
- 轻视文档的作用和软件运行期的维护。
- 缺少开发方法学指导。

软件危机产生的主客观因素如图 1-2 所示。

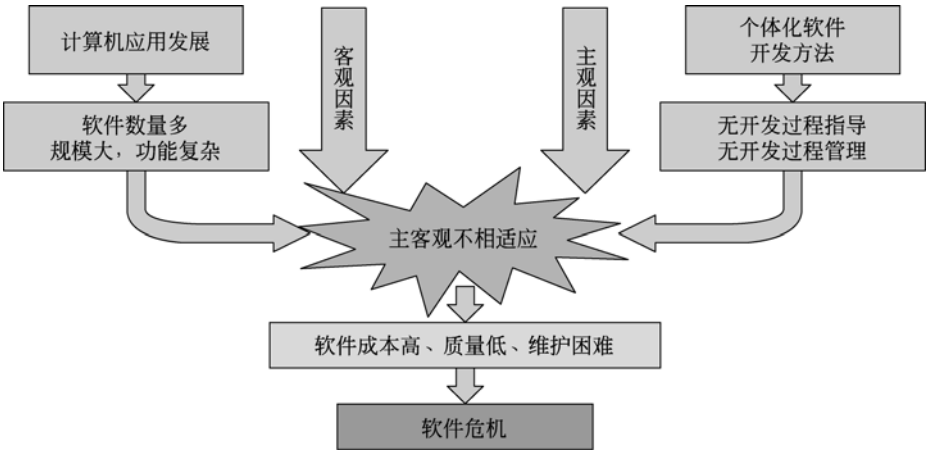


图 1-2　软件危机产生的原因

1.2.3　软件危机的表现

（1）软件开发进度难以预测、开发成本难以控制，导致超预算、超时。
（2）产品功能难以满足用户需求。
（3）软件产品质量无法保证。
（4）软件缺少相应的文档资料，维护困难。
（5）软件成本超过硬件成本。
（6）软件开发生产率提高的速度远远跟不上计算机应用迅速普及深入的趋势。

1.3　解决软件危机的途径

软件危机产生的根本原因就是没有把软件的加工当作工程来管理，为了解决软件危机，人们借鉴了其他产业的经验提出：用工程化的理论、方法和技术来管理软件开发过程，

从管理和技术两方面对软件开发进行规范化和理论研究，即产生了软件工程学科。软件工程的基本内容如图 1-3 所示。

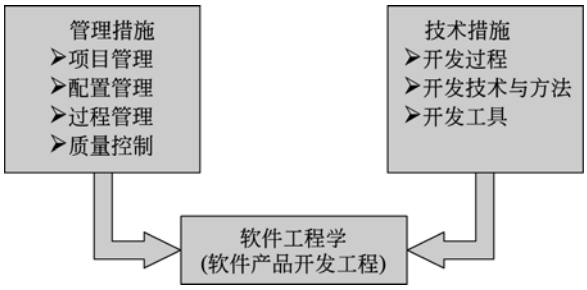


图 1-3　软件工程基本内容

1.4　软件工程的定义

“软件工程”对照的英文可以有：Software Engineering；Software Project；Software Engineer。不同的权威机构、不同的权威人士、不同的权威教科书对软件工程给出了表述语句不同、意义相近的定义。

- Fritz Bauer 在 NATO 会议上给出的定义：“软件工程是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确立和使用的健全的工程原理（方法）。”
- IEEE【IEE83】给出的软件工程定义：“软件工程是开发、运行、维护和修复软件的系统方法。”
- IEEE【IEE93】给出了一个更加综合的定义：“将系统化的、规范的、可度量的方法应用于软件的开发、运行和维护的过程，即将工程化应用于软件中。”
- 美国的软件工程学家格莱斯对软件工程下的定义是：“软件工程是用一系列的工具来制造和维护一个有关现实世界的自动解决方案。”
- Fairley 给出的软件工程的定义为：软件工程学为在成本限额以内按时完成开发和修改软件产品所需的系统生产和维护的技术与管理的科学。

国内教科书关于软件工程定义普遍接受的描述是：**软件工程是指应用计算机科学、数学及管理科学等原理，以工程化的原则和方法来解决软件问题的工程。其目的是提高软件生产率、提高软件质量、降低软件成本、满足用户需求。**

张海藩《软件工程导论》各版中一直沿用如下定义：**软件工程是指导计算机软件开发和维护的一门工程学科。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，以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。**

不管哪个定义，都只是文字上的区别，没有原则上的分歧，读者没有必要对每个描述是否合理进行严格甄别，重要的是通过阅读上述软件工程的定义初步理解软件工程的作用。

（1）软件工程是指导对计算机软件产品进行计划、开发和维护的工程学科。

（2）软件工程促进软件行业从无序到有序，要按照工业化生产过程进行软件开发

活动，即任何生产活动都要按照目标化、规范化、文档化、标准化进行，这就是工程化。

- 目标化。工程化的目标是在满足质量的前提下使投入产出的效益最大化。
- 规范化。有明确的工作阶段、内容和步骤。
- 文档化。有详细具体的工作规范化文档。
- 标准化。有明确的质量评价体系和标准。

1.5 软件工程的内容

软件工程是围绕软件开发的一门交叉学科：软件工程＝计算机科学+工程学+管理学。

1.5.1 从计算机科学视角看软件工程

从事软件开发，应具有计算机科学方面的理论基础、逻辑思维和知识体系，如离散数学、数据结构、算法分析、程序设计等专业基础课程可以培养思维方式从连续向离散转变，培养程序员的基本功。

但这些课程只是培养个人单打独斗的技术能力，没有培养工程化思维，也就是缺乏对软件加工的生产过程、资源计划、质量保证、生产效率、工程效益、风险控制以及团队合作等工程意识的培养，因此获得计算机科学方面的基础训练是步入软件工程的基础。

1. 软件工程的知识体系

IEEE 的软件工程知识体系指南（SWEBOK）中界定了软件工程的 10 个知识领域（Knowledge Areas，KAs），这 10 个知识领域的每个领域还包括很多子领域。

- 软件工程过程（Software Engineering Process）。
- 软件工程工具和方法（Software Engineering Tools and Methods）。
- 软件需求（Software Requirement）。
- 软件设计（Software Design）。
- 软件构造（Software Construction）。
- 软件测试（Software Testing）。
- 软件维护（Software Maintenance）。
- 软件配置管理（Software Configuration Management）。
- 软件工程管理（Software Engineering Management）。
- 软件质量（Software Quality）。

2. 软件工程三要素

从计算机科学观点看，软件工程是一种层次化技术，分别是过程、方法和工具，如图 1-4 所示。

（1）软件工程必须以有组织的质量保证为基础，进行全面质量管理，不断地改进过程使软件工程方法走向成熟。

（2）“过程”是软件产品加工所经历的一系列有组织的活动，从而能够合理、高质量和及时开发出计算机软件。过程定义的活动集合及其序列见第 2 章。

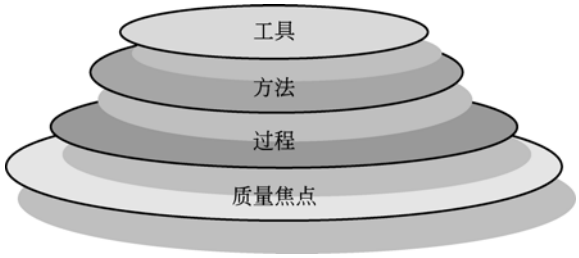


图 1-4　软件工程三要素：过程、方法和工具

（3）“方法”为软件开发提供“如何做”的技术，它涵盖了项目计划、需求分析、系统设计、程序实现、测试与维护等一系列活动的做法。如开发方法经历了面向结构、面向对象、面向组件的发展过程，还有项目管理中的估算、度量、计划等管理方法。目前软件工程多以介绍方法为主，本书从第 3 章开始逐步介绍开发过程所用到的方法。

（4）“工具”为过程和方法提供自动或半自动的支持。这些软件工具既包括软件也有硬件。软件工具包括编程、建模、管理等开发工具。通过网络把这些软件工具集成起来搭建一个支持团队开发的平台，称之为计算机辅助软件工程(Computer Aided Software Engineering，CASE)。CASE 集成了软件、硬件和一个存放开发过程信息的软件工程数据库，形成了一个软件工程环境。

软件工程所包含的内容不是一成不变的，随着人们对软件开发和生产的理解，应用发展的眼光看待它。

3. 转变学习思维

从计算机科学进入到软件工程的学习，要转变学习思维。

（1）将模块的算法分析与程序设计的思维定式提升到系统分析与设计层面。

（2）将只定位程序员的训练提升到系统工程师培养层面。

（3）将只关心符合计算机逻辑提升到符合工程规范的层面。

1.5.2　从工程视角看软件工程

工程与科学研究不同，工程讲究投入产出最大化。作为工程一定要追求多快好省的“目标”、按有序的活动“过程”进行、遵循一定的“原则”。软件工程也不例外，它的目标、过程和原则构成了三维全景框架，如图 1-5 所示。

（1）软件工程的目标：追求高质量、高效率、高效益。

在给定成本、进度的前提下，开发出具有可修改性、有效性、可靠性、可理解性、可维护性、可重用性、可适应性、可移植性、可追踪性和可互操作性并满足用户需要的软件产品。图 1-5 中用可用性（符合功能要求）、正确性（符合质量要求）、合算性（达到经济效益）

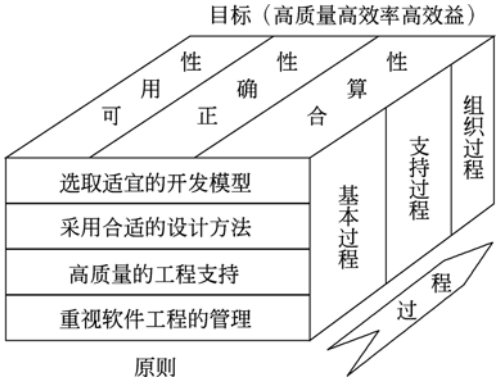


图 1-5　软件工程的三维全景图

进行高度概括，也就是产品开发要高质量、高效率、高效益。

（2）实现目标的过程：完成产品加工的过程。

产品的加工要通过一系列的技术和管理过程（详见第 2 章）以达到目标，它包括以下内容。

① 基本过程：获取、供应、开发、操作、维护。

② 支持过程：文档、质量、验证、评审。

③ 组织过程：改进、培训。

（3）进行过程应遵守的原则：行进中的轨道约束。

原则是指导过程进行中所必须遵循的大政方针。软件工程提出的 4 项基本原则，全面指导软件工程的过程、方法和工具的使用以及对过程的管理。不但指导个别的项目开发，也指导整个团队的建设。

① 选取适宜开发范型。是指选取和运用合适的软件过程、开发方法、体系结构等方向性模式，它们是相互支撑、相互制约的。选取它们就是考虑软件生存的生态环境，及时生产出满足用户需求的软件。使得软件开发处在易开发、易维护、易扩展、易复用、易管理的生产状态之中。现代技术提供了各种优秀的范型，如 RUP 与 XP 软件过程、B-S 体系结构、Web 分层结构、面向组件的开发方法、各种架构与框架等范型。

② 采用合适的设计方法。软件设计中孜孜不倦追求的是松耦合。合适的设计方法就是贯彻模块的独立性原则，达到松耦合。现代开发技术已经提供给人们很多好的设计选择，如面向对象的 GOF、面向分层的 MVC、面向方面的 AOP 等设计模式都是追求松耦合的结果。

③ 提供高质量的工程支持。“工欲善其事，必先利其器”，这是促进技术进步的源动力，是各种工程共同追求的因素，它对提高产品质量与工作效率有直接效用。近年来，软件开发工具与环境有很大改善，如集成开发工具、团队开发环境、配置管理、测试工具、建模工具等可供选择的产品很多。

④ 重视开发过程的管理。过去人们的关注点更多地集中在方法和工具方面，随着人们对能力成熟度模型 CMM 和软件过程改进 SPI 的认识的提高，现在更加重视对开发过程的管理。软件过程的管理，直接影响可用资源的有效利用、生产满足目标的软件产品、提高软件组织的生产能力等问题。因此，仅当软件过程得以有效管理时，才能实现有效的软件工程。

1.5.3 从管理视角看软件工程

软件工程除了在技术层面上研究它的过程、方法和工具外，本身充满着管理的学问，这些管理方面的知识正是理工类学科培养所缺乏的。又由于软件是一种抽象的逻辑产品，是人的高智慧劳动，没有外在的形状和显著的物理加工，其规模难以估算、劳动效率和产品质量难以评价、加工过程难以控制，因此软件工程管理与物理加工的传统管理既具有共性又具有特殊性，既要借鉴传统工程管理规范又要发展新的理论和实践，所以对软件工程的管理是一个不断发展和完善的新型管理学科。

软件工程管理的主要形式是项目管理。随着软件开发规模的扩大、创新技术的不断引入以及软件产业的形成，人们越来越意识到软件过程管理的重要性，管理学的思想逐渐融

入软件开发过程，项目开发的管理日益受到重视。软件项目管理就是运用一系列的知识、技能、工具和技术，在软件开发的活动中有效地掌控资源，对项目时间、质量和成本进行管理，如项目计划、团队管理、质量管理、过程管理、过程改进、配置管理、风险管理等。这些管理是现代软件工程重点关注的内容，本书第 10 章集中介绍了项目管理，但有些内容只是散落在相关章节里做了简单的介绍。

1.5.4　从基本原理视角看软件工程

自从 1968 年提出“软件工程”这一术语以来，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。美国著名的软件工程专家巴利·玻姆（Barry Boehm）综合了这些专家的意见，并总结了美国天合公司（TRW）多年的软件开发经验，于 1983 年提出了软件工程的 7 条基本原理。

玻姆认为，这 7 条原理是确保软件产品质量和开发效率的原理的最小集合，它们是相互独立的，是缺一不可的最小集合；同时，它们又是相当完备的。人们当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 多条软件工程准则都可以由这 7 条原理任意组合蕴含或派生。下面简要介绍软件工程的这 7 条原理。

1. 用分阶段的生命周期计划严格管理

把软件生命周期分成若干阶段，并相应制定出切实可行的计划是为了将软件开发复杂性变得容易控制和管理。统计表明，50%以上的失败项目是由于计划不周而造成的。在软件开发与维护的漫长生命周期中，不同阶段需要完成性质各异的工作，每个阶段要严格按照计划对软件的开发和维护进行管理。Boehm 认为，在整个软件生命周期中应指定并严格执行 6 类计划：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、运行维护计划。

2. 坚持进行阶段评审

有了阶段的划分和阶段目标，要对阶段性工作进行评审，也就是实行阶段控制。根据 Boehm 等人的统计，设计错误占软件错误的 63%，编码仅占 37%；错误发现得越晚错误的传播和放大越严重，改正它要付出的代价就越大，要差 2～3 个数量级。 因此，软件的质量保证工作不能等到编码结束之后再进行，应坚持进行严格的阶段评审，以便尽早发现错误。

3. 实行严格的产品控制

有了前面两条才能在开发和维护环节对产品的成本、进度、质量、变更进行控制。在软件开发过程中“最怕”的是需求的变更，但需求改变又是难免的。如果执行产品变更控制技术，一切有关修改软件的建议，特别是涉及对基线配置的修改建议，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. 采纳现代程序设计技术

从早期的结构化软件开发技术，到最近的面向对象技术以至今天的面向架构、面向服务的组件技术，从第一、第二代语言到第四代语言，人们已经充分认识到：先进的技术既可以提高软件开发的效率，又可以减少软件维护的成本。

5. 结果应能清楚地审查

软件是一种看不见、摸不着的逻辑产品。对软件开发人员的工作进展情况进行监测可见性差、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，从而使所得到的结果能够清楚地审查。

6. 开发小组的人员应少而精

开发人员的素质和数量是影响软件质量和开发效率的重要因素，应该少而精。这一条基于两点原因：高素质开发人员的效率比低素质开发人员的效率要高几倍到几十倍，开发工作中犯的错误也要少得多；当开发小组为 N 人时，可能的通信信道为 $N(N-1)/2$，可见，随着人数 N 的增大，通信开销将急剧增大。软件项目管理一章介绍的软件估算模型（COCOMO）及人员配备规则（10.5.2 节）都支持这一理论。

7. 承认不断改进软件工程实践的必要性

遵从上述 6 条基本原理，就能够较好地实现软件的工程化生产。但是，它们只是对现有经验的总结和归纳，并不能保证紧跟技术不断前进的步伐。因此，Boehm 提出应把承认不断改进软件工程实践的必要性作为软件工程的第 7 条原理。根据这条原理，不仅要积极采纳新的软件开发技术，还要注意不断总结经验，收集进度和消耗等数据，进行出错类型和问题报告统计。这些数据既可以用来评估新的软件技术的效果，也可以用来指明必须着重注意的问题和应该优先进行研究的工具和技术。这一原理后来发展为“软件过程改进”，详见 2.6 节。

1.6 软件发展的新阶段与新问题

1.6.1 软件发展的第 4 阶段和第 5 阶段

在软件发展的前三个阶段暴发了软件危机。危机并不可怕，危机是打破现有平衡促进事物发展的必然产物，每次新的危机出现就是一次新的发展契机。寻求解决软件危机促进了软件工程的发展，带来了软件产业欣欣向荣的第 4 个阶段。

计算机软件发展的第 4 阶段是 20 世纪末到 21 世纪初（1990—2000 年）。这个时代的出现以个人计算机（PC）的出现为时代标志。1981 年，IBM 推出了 IBM PC 迎接新的软件时代开始，微软是这个时代最成功和最有影响力的代表。这个时期是强大的桌面系统和计算机网络迅猛发展的时期，使得计算机应用从企事业普及到家庭，计算机体系结构由中央主机控制方式变为客户-服务器方式，专家系统和人工智能软件终于走出实验室进入了实际应用，虚拟现实和多媒体系统改变了与最终用户的通信方式，出现了并行计算和网络计算的研究，面向对象技术在许多领域迅速取代了传统软件开发方法。软件市场形成产业化、规模化。这个时代可以称为“客户大众市场软件时代”。

进入 21 世纪，随着经济全球化的发展趋势和全球化市场竞争压力的增加，企业级软件需求发生了更大的变化。如电子商务（Electronic Commerce，EC）、供应链管理（Supply Chain Management，SCM）、企业资源计划（Enterprise Resource Planning，ERP）、客户关

系管理（Customer Relationship Management，CRM）、网络协同（Web Collaboration）等软件需要跨部门、跨区域、跨平台，软件规模大、功能复杂、技术综合性强。面对这样的大型复杂软件的开发，暴露了新的危机现象，对企业提出了新的挑战。

企业发展需要更多的业务灵活性和创新能力，企业永远处在变化中，“变化是持续的、永恒的”，环境在变、需求在变、技术在变、架构在变，这就要求为企业打造的软件应当适应企业外部环境、经营方式、流程再造所带来的变化。然而，以往硬编码形成的软件是一种刚性结构，而且是采用一次性开发并交付的定制方式，交付后的软件不能满足不断膨胀、不断变化的需求，企业用户只能通过不断开发新应用、扩展现有应用程序来艰难地支撑其不断扩张的业务需求。有的软件交付后很快成为遗留系统，甚至于没开发完就已经落后于新的需求和 IT 环境了。传统开发技术的软件维护困局日渐凸显，这就是客观发展逐步暴露出的软件新问题，即软件不适应变化，或称新的软件危机。解决这个危机也就发展到了软件开发的第 5 阶段。第 5 阶段从 21 世纪开始到现在仍在进行中。第 5 阶段要解决的问题是：软件结构是柔性的，模块之间是松耦合的，做到开发过程分散解决、逐步集成，维护上可任意裁剪，满足分布式计算方面追求的伸缩性、安全性、异构和互通性方面的要求。该阶段又经历了面向架构、面向服务和软件即服务的发展过程。

1.6.2　以面向对象为基础的面向架构技术

由顶级软件供应商提供的企业架构技术和中间件产品，为在企业级软件中应用面向对象技术提供了完整的解决方案和技术规范，如微软的.NET 架构和 Sun 的 J2EE 架构。企业架构为编程者提供组件，然后将组件发布到管理组件运行的容器上，为企业级软件开发带来极大的方便，企业级软件的开发效率、质量、维护性、成功率得到空前的提高。面向架构技术也称面向组件技术，架构提供了分层结构、不同层采用不同的组件、执行不同的职能，因此可以做到静动态代码的分离；应用架构也可对组件进行静态柔性装配，动态管理对象的生命期和依赖关系。架构对软件适应变化、增加柔性、改善维护性提供了前所未有的优势，把基于面向对象的 Web 开发推向简化、实用和规范。目前在业界正如火如荼地采用架构技术进行企业级软件开发。面向架构的开发方法是本书的重点，详见第 11 章。

面向组件的企业架构可以提供柔性的组件装配模式，但在解决面向业务的粗粒度、跨平台松耦合的业务集成方面存在的瓶颈问题没有得到有效的解决，新一代面向服务架构 SOA 的诞生将会敲开通往业务组件化、集成化的大门。

1.6.3　以业务单元为基础的面向服务架构 SOA

面向服务架构（Service-Oriented Architecture，SOA）是 20 世纪末提出、21 世纪初被逐渐接受和实现的概念。SOA 主要用于解决传统对象模型中无法解决的异构和耦合问题。SOA 是一个组件模型，它的封装单元可以是应用程序级，每个单元可执行不同业务功能，每个业务单元称为服务，通过这些服务之间定义良好的接口和契约联系起来。接口是采用中立的方式进行定义的，它独立于实现服务的硬件平台、操作系统和编程语言。这使得构建在各种各样的系统中的服务可以以一种统一和通用的方式进行交互。这种具有中立的接口定义（没有强制绑定到特定的实现上）的特征即为服务之间的松耦合。因此 SOA 是“抽

象、松散耦合和粗粒度”的软件架构，它可以根据需求通过网络对松散耦合的粗粒度应用组件进行分布式部署、组合和使用。目前，SOA 已从概念走向实现，其中 Web 服务（Web Service）就是 SOA 的一种实现方式。

SOA 的中心思想就是使得企业软件应用摆脱面向技术的束缚，演变成面向业务整合、面向业务流程的解决方案，轻松应对企业发展变化的需要。有了 SOA 可以使得软件结构柔性可伸缩、跨异构平台；开发过程可以分散开发、逐步集成；在维护上，组件可任意插拔、功能可任意裁剪、业务流程可随意改造。因此应用 SOA 可以有效整合和重用现有的应用系统和各种资源，对各种服务进行服务组件化，并基于服务组件实现各种新的业务应用的快速组装，帮助企业更好地应对业务的灵活性要求，详见第 11.4 节。

与 SOA 相关的是另外一种服务架构 SCA（Service Component Architecture，服务组件架构）。SCA 是 SOA 的一种更好的实现方式，可以从细粒度 Class 级组件到 Web Service 粗粒度组件进行组装和通信，详见第 11.4.3 节。

1.6.4 以软件作为服务的应用模式

这里提出的“服务”与 SOA 里的“服务”是两个不同层面的概念。SOA 的“服务”是组成软件系统的开发模式，属于技术层面；而这里的“服务”是软件的应用模式，属于商业层面。这里的“服务”从其目标看已超出目前软件工程的讨论范围，但却勾画出软件工程的美好愿景，而且正是由于 SOA 等技术的发展催生了“软件作为服务”的实现，因此这里将其作为新知识的扩展为读者做简要介绍。

Sun 公司提出了“网络就是计算机”（The Network is the Computer™）这一短语。互联网技术的成熟，将软件从计算机迁移到网络，软件不是通过销售渠道获得，而是通过网络获得或组成，企业不用找人来开发软件，而是在网络上应用合适的软件，这就是 21 世纪初提出的一种完全创新的软件应用模式 SaaS（Software as a Service，软件即服务）。

SaaS 是一种通过 Internet 应用软件的模式，厂商将应用软件统一部署在自己的服务器上，客户可以根据自己的实际需求，通过互联网向厂商订购所需的应用软件服务，按订购的服务多少和时间长短向厂商支付费用，并通过互联网获得厂商提供的服务。用户不用再购买软件，而改向提供商租用基于 Web 的软件，来管理企业经营活动，且无须对软件进行维护，服务提供商会全权管理和维护软件。软件厂商在向客户提供互联网应用的同时，也提供了软件的离线操作和本地数据存储，让用户随时随地都可以使用其订购的软件和服务。对于许多小型企业来说，SaaS 是采用先进技术的最好途径，它消除了企业购买、构建和维护基础设施和应用程序的需要。在这种模式下，客户不再像传统模式那样花费大量投资用于硬件、软件、人员，而只需要支出一定的租赁服务费用，通过互联网便可以享受到相应的硬件、软件和维护服务，享有软件使用权和不断升级，这是网络应用最具效益的运营模式。

基于 SaaS 概念，促进了计算虚拟化（Virtualization）和云（Cloud）的迅猛发展。目前关于虚拟化和云之间的关系尚有争论。尽管两者不是相互绑定，甚至有人将两者作为两个问题讨论，但要讨论其中一个话题就必然涉及另一个，它们纠结在一起很难分开。虚拟化平台 VMware 设计专家大卫·康弗瑞认为：“虚拟化为云计算提供了很好的底层技术平台，而云计算则是最终产品。”即云是一种基于互联网的商业计算模型，虚拟化是实现这个商业

模式的底层技术。

虚拟化打破了物理结构之间的界线，可以高效利用硬件、软件、数据、网络、存储等计算资源。虚拟化有许多不同的类型，包括机器虚拟化、表示层虚拟化、应用程序虚拟化、存储虚拟化、网络虚拟化。

云由 SOA、虚拟化、Web 2.0 等热点技术相融合发展而来。云就是网络上的一些大型服务器集群，为客户提供计算服务、存储服务等无须客户管理和维护的虚拟计算资源。云指云计算（Cloud Computing）和云服务（Cloud Service），计算是手段、服务是目的。通常说的云计算指的是云服务运营商通过分布式计算和虚拟化技术搭建数据中心、网络服务器集群或超级计算机，以按需租用/计费方式向技术开发者或者企业客户提供数据存储、软件服务、硬件租借、计算分析等不同类型的服务。对云服务运营商来说，它的职责是如何构建和管理云计算平台，对客户来说就是用好云服务。运营商可以构建不同层面的虚拟化，通过云为不同类型用户提供不同层面的服务，如图 1-6 所示。

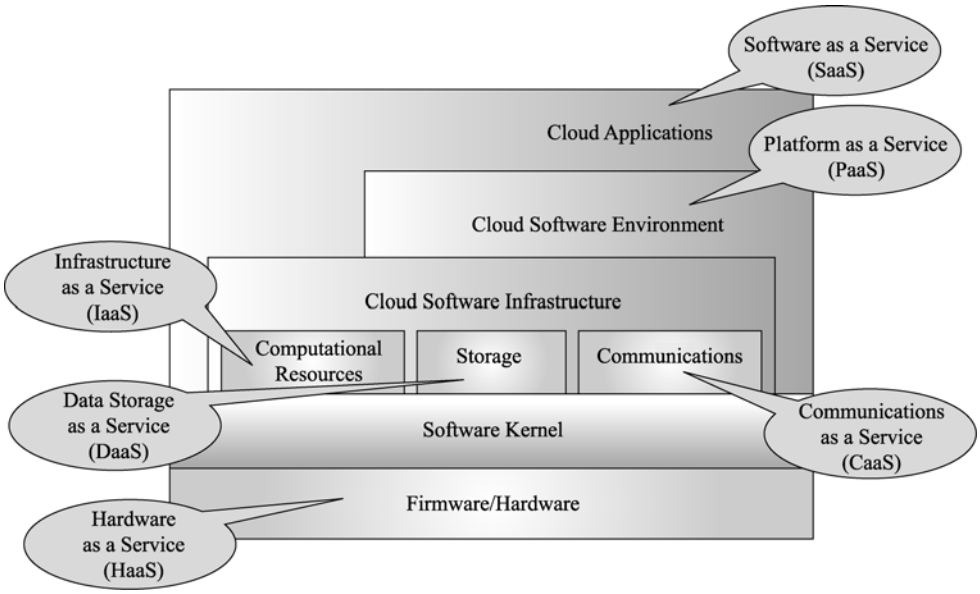


图 1-6　云计算提供的服务层次

（1）最下层是基础设施，包括硬件、服务器等物理资源，提供这类服务称为基础设施服务（Infrastructure as a Service，IaaS）。消费者可以将自己系统所用的操作系统、各种各样的中间件产品以及应用程序送到一个虚拟机上运行在云中。对于建有数据中心的客户，使用 IaaS 即利用云中的 CPU、内存、存储以及网络，能够大幅度提高资源使用效率从而降低成本。

（2）第二层是中间平台服务（Platform as a Service，PaaS），它将各个阶段的软件开发集成环境（建模、编程语言和工具、测试、维护、配置管理等）和基础平台（解释器、Web 层和持久层容器等）作为服务提供给软件开发企业。也可以提供领域基础平台直接供应给终端用户，客户通过使用 PaaS 可以将自己的应用程序打包部署以及运行在云中，并且可以通过 API 调用 PaaS 提供的各种共享服务，比如像数据持久化服务以及应用消息服务。云

中的 PaaS 具有更好的伸缩性，云用户只专注于自己的应用，而不需要关注云的底层支撑。PaaS 特别适合以自己的应用为中心的中小企业，可以大幅度降低甚至省掉他们采购以及维护中间件产品的费用。这个服务层面类似于早期出现的应用服务托管（Application Service Provider，ASP）。

（3）最上面一层是软件即服务（Software as a Service，SaaS），通过互联网为用户提供了具体的业务功能。如 Google Apps 提供的云 Office 可取代本地安装的 MS Office，这类属于通用程序和实用工具；还有的企业使用如 CRM、ERP 等专业云；还有专用服务如 IBM 提供的在线会议。SaaS 特别适合于那些需要直接使用某种特定应用的客户，这类客户不需要开发任何应用程序，只需要根据自己的云应用进行付费即可。

云计算的消费者如果是利用云的并行计算能力来完成大型科学计算，这就是云计算的前辈“网格计算（Grid Computing）”。云计算的发展目标主要还是用在企业的管理和商务活动上，因此下面介绍的云服务模式都是以企业的口气来讨论的。云服务模式分为公用云、专用云和混合云。

公用云由第三方运行，不同客户使用的应用程序可能会在云的服务器、存储系统和网络上混合在一起，如图 1-6 所示。公用云通常在远离客户建筑物的地方托管。企业可以建立自己的虚拟化平台构建专用云，专用云可由公司自己的 IT 机构也可由云提供商进行构建。专用云是为单一客户单独使用而构建的，因而提供对数据、安全性和服务质量的最有效控制。专用云拥有自己的基础设施，当然也可以基于公用云 IaaS 将它们部署在一个主机托管场所。如果把公用云与专用云结合在一起就称为混合云。混合云有助于提供按需的、外部供应的扩展，利用公用云的资源来扩充专用云的能力可缓冲工作负荷的快速波动。

云计算及 SaaS 已不是神话，正脚踏实地扑面而来。目前云计算市场竞争激烈，IBM、微软、雅虎、亚马逊、EMC、Google、Oracle（包括被收购的 BEA 和 Sun）等顶级 IT 开发商都推出了可打造云计算平台的产品或云服务。在企业级软件领域，产品技术和市场以美国 Salesforce（http://www.salesforce.com/）为领头羊。国内类似的厂商以八百客（http://800app.com/）、沃力森（http://www.xtools.cn/）为主，主要开发 CRM、ERP 等在线应用。用友、金蝶等老牌管理软件厂商也推出在线财务 SaaS 产品。国际上其他大型软件企业中，微软提出了 Software+SaaS 的模式；谷歌推出了与微软 Office 竞争的 Google Apps；Oracle 在收购 Sieble 后推出 Oracle On-demand；SAP 推出了和传统 SaaS 的杂交（Hybrid）模式。

1.7 软件工程的演变发展

1.7.1 从软件技术到软件工程

从表 1-1 可以看到软件工程并不是与软件技术同步发展的。计算机生态系统由硬件环境、网络环境、软件环境 （开发语言/建模语言/开发工具）、软件工程组成，软件工程处在进化链的顶端。软件工程与其他生态系统的进化同样遵循链式效应，即顶端的发展总是由底层环境的进化推动的，也就是软件工程即软件开发方法的发展总是滞后于技术的

发展，通常是在软件的某项语言、技术或工具进入稳定期后，才会催生软件工程相应新的思想出现。

20 世纪 50 年代到 60 年代，先后出现了机器语言、汇编语言、过程性高级语言（FORTRAN、COBOL、Lisp）[8]，这时期的编程是依靠个人的理解，没有可遵循的方法学和规范。20 世纪 70 年代，过程语言继续发展，出现了 ALGOL、BASIC、Pascal、C 和 Ada 等语言，与此同时 Simul、Smalltalk、Eiffel 等面向对象语言开始应用，这时期才诞生了结构化的程序设计方法学，人们开始应用结构化（顺序、分支、循环）思想来控制程序。进入 20 世纪 80 年代，随着混合型面向对象语言（在过程式语言及其他语言中加入类、继承等成分）C++的广泛应用，有关模块划分、数据隐藏等模块独立性思想被强调和接受。而软件实现是对客观世界进行抽象与细化的理念刚刚被认识，软件开发要经过获取、分析、设计、实现、发布、维护过程的软件开发方法学才刚刚起步，瀑布模型刚刚显现，软件工程才开始被重视。20 世纪 90 年代，当面向对象技术与网络紧密结合，软件复用和组件技术成为主流应用时，面向对象的开发方法 OOA 和 OOD 才趋于普及和成熟，统一和敏捷过程才开始出现，人们开始注重软件过程改善。从过去的发展结果看，软件技术和软件工程发展之间的滞后间隔大约 5～10 年，但可以预见，未来这种间隔会逐步缩短，滞后期有可能缩短到 2～3 年。这种链式的延迟性的简单归纳如表 1-2 所示。

表 1-2　软件工程与软件技术的时滞

年　　代	开发语言与技术	软 件 特 征	软件工程进化
20 世纪 50—60 年代	机器语言； 汇编语言； 过程性高级语言	个性化	无
20 世纪 60—70 年代	过程性高级语言继续发展； 面向对象语言出现	结构化程序	程序设计方法学
20 世纪 70—80 年代	面向过程继续应用； 面向对象技术广泛应用	功能模块化程序	软件工程=软件开发方法学=过程+管理，遵循古典软件生存周期模型
20 世纪 80—90 年代	分布式计算=面向对象+网络技术	类模块及组件化程序	项目管理、质量管理、过程改善、配置管理受重视；遵循统一过程和敏捷开发模型
20 世纪末—21 世纪初	企业架构技术广泛应用； 面向服务从概念走向应用； 虚拟化和云计算起步	组件服务化；跨平台、组装式、柔性结构	生产过程仍沿用成熟的面向对象开发方法；模型驱动等新方法处在研究阶段

1.7.2　从传统软件工程到现代软件工程

有人把软件工程的进化分为传统软件工程和现代软件工程，这种划分没有严格界限，因为事物的发展是交错、延伸、相对的。但这种划分对了解软件工程的进化、指导软件工程的学习还是有意义的。

1. 传统软件工程

传统软件工程是基于结构化的软件开发方法，尽管比较原始，但在观念突破、解决软

件危机方面是个良好的开始，有许多实际作用。

（1）遵循阶段控制、降低难度、控制风险的理论，提出了软件开发经过计划时期、开发时期、运行维护时期的软件生命周期模型，建立了软件开发以分析→设计→编码→测试→维护为活动顺序的瀑布模型，在此模型的基础上发展出增量模型、快速原型法、螺旋模型等改进模型。

（2）改变不做需求分析就开始编程的习惯，注重软件文档在开发和维护中的作用，树立软件是程序+文档的观念。

（3）按照自顶向下、逐步求精的原理贯穿结构化的开发过程，分析阶段采用面向数据的结构化功能分析方法，设计阶段采用面向模块、面向过程的结构化程序设计方法。

（4）软件规模估算的方法和模型使得一塌糊涂的软件项目的资源规划、进度计划和管理变得理性化。

传统软件工程提出的软件开发方法是基于数据驱动的，在分析时以数据和数据流进行功能分析和分解，系统由功能模块组成，有效地解决了系统复杂性。但传统软件工程有一些与生俱来的局限。

（1）功能分析不是基于问题域的，不是基于客观事物原貌作为基本单元，而是基于数据的（面向数据流或面向数据结构），因此需求分析结果很难做到完整、客观、一致、可靠。

（2）从“分析模型”到“设计模型”是面目全非的表达方法，这种映射规则很难做到理解上的一致，也很难保证前后两个模型的一致。

（3）面向过程技术的限制，难以建立通用性强的程序库，模块的重用性差。

（4）用户需求的变化会造成软件结构发生较大变动甚至是颠覆性的改变，用户需求的变化对软件开发周期和开发成本带来不确定性。

2. 现代软件工程

现代软件工程是以面向对象（Object Oriented）技术为标志，体现为开发技术、开发方法、管理理念的全面创新和应用。

1）面向对象技术从语言、工具、方法到过程的发展

（1）多种面向对象编程语言满足各种应用程序的需求。从 C++的广泛应用开始，先后出现的用于桌面、Web、可视化、嵌入式、移动设备等面向对象语言如雨后春笋、生机勃发。TIOBE 官网于与 2010 年 8 月发布编程语言排行榜[9]，如表 1-3 所示。表中反映了市场占有率和位置变动情况。最前的 Java、C 和 C++的排名一直是比较稳定的，Web 开发语言 C#、PHP、Python 和 Ruby 也占有一定的市场份额，值得关注的是网络平台 Go 语言和移动平台 Objective-C 语言正日益崛起。

（2）面向对象的开发方法 OOA、OOD 从百家争鸣走向用例驱动的统一方法。统一建模语言 UML 诞生及建模工具的使用、有效地支持了 OOA 和 OOD。OOA 即为面向领域、面向客观的分析方法，与面向数据的结构化分析方法有质的区别，OOA 使得不同教育背景的领域用户和软件分析者之间容易交流，不同分析者的结论分散性有所收敛，分析结果的效用性显著提高。

（3）融进了增量、重叠、迭代、里程碑等先进思想的统一软件过程模型（Rational Unified Process，RUP），为开发团队提供了可裁剪的软件过程模型；基于敏捷开发思想的极限编程（eXtreme Programming，XP）使开发团队灵活应对需求经常发生变化的项目。

表 1-3　TIOBE 2010 年 8 月公布的程序设计语言排行榜

Position Aug 2010	Position Aug 2009	Delta in Position	Programming Language	Ratings Jul 2010	Delta Jul 2009	Status
1	1	=	Java	17.994%	−1.53%	A
2	2	=	C	17.866%	+0.65%	A
3	3	=	C++	9.658%	−0.84%	A
4	4	=	PHP	9.180%	−0.21%	A
5	5	=	(Visual)Basic	5.413%	−3.07%	A
6	7	↑	C#	4.986%	+0.54%	A
7	6	↓	Python	4.223%	−0.27%	A
8	8	=	Perl	3.427%	−0.60%	A
9	19	↑↑↑↑↑↑↑↑↑↑	Objective-C	3.150%	+2.54%	A
10	11	↑	Delphi	2.428%	+0.09%	A
11	9	↓↓	JavaScript	2.401%	−0.41%	A
12	10	↓↓	Ruby	1.979%	−0.51%	A
13	12	↓	PL/SQL	0.757%	−0.23%	A
14	13	↓	SAS	0.715%	−0.10%	A
15	20	↑↑↑↑↑	MATLAB	0.627%	+0.07%	B
16	18	↑↑	Lisp/Scheme/Clojure	0.626%	0.00%	B
17	16	↓	Pascal	0.622%	−0.05%	B
18	15	↓↓↓	ABAP	0.616%	−0.12%	B
19	14	↓↓↓↓↓	RPG(OS/400)	0.606%	−0.15%	B
20	-	↑↑↑↑↑↑↑↑↑↑	Go	0.603%	0.00%	B

2）融入更多、更新的管理理念和手段

（1）SPI 与 CMM 是质量保证的重要手段。依靠技术进步来提高软件质量已不是唯一途径，人们逐渐认识到软件工程完善的解决方案＝技术提高+过程提高。软件过程是指软件组织建立一套完整的产品开发过程框架，规范过程中的活动、任务和执行顺序，指导具体的工程开发。企业的软件过程不是一成不变的，应参照成长模型（评估标准）具有自我提高的能力，这就是软件过程改善（Software Process Improvement，SPI），企业通过不断地改善软件过程，实现成本、进度、功能和产品质量等目标。参照的成长模型就是软件能力成熟度模型（Capability Maturity Model For Software，CMM）。

（2）软件配置管理（Software Configuration Management，SCM）逐步受到重视。配置管理是基于加利福尼亚大学的 Leon Presser 教授在 20 世纪 70 年代提出的软件变更和配置的概念，逐渐在国内外成熟的软件企业中得到重视和普及的一门软件项目管理技术。配置管理也是 CMM 2 级的一个关键过程域。配置管理主要解决在团队开发环境下由于需求变更或版本升级所造成的文档管理混乱的局面，记录软件的变更，重用软件的配置项，使软件的变更管理变得有序，新的扩展工作具有延续性和可追溯性。

3）架构技术的应用改变了开发模式并全面提升软件质量

（1）J2EE 和.NET 等企业架构在业界得到广泛的应用。企业架构可以减轻高层设计和管理的麻烦，使软件开发者集中关注领域问题，使软件的开发效率、质量、成功率空前提高。

（2）企业架构使得面向对象技术发挥得淋漓尽致，开发者真正享受到面向对象所带来的迷人之处。诸如 MVC 设计模式、面向方面编程（Aspect Oriented Programming，AOP）、对象的控制反转（Inversion of Control，IoC）和依赖注入（Dependency Injection，DI）等技术的应用，不仅把面向对象的开发方法提高到一个新的境界，更能使开发者领略到软件开发哲学的魅力。

（3）SOA 和云计算将促使现代软件工程更加现代化。SOA、虚拟化、云计算等新一代技术正在蓬勃兴起，这些技术为企业级软件的构造提供了更优秀的解决方案，对于企业级软件开发模型带来更加翻天覆地的变化，在不久的将来会成为主流应用技术。

面对技术上的变化，尤其是架构技术的应用和深入发展，原来基础上的面向对象软件工程是否能承载下这些变化？用原来的需求分析方法、建模方法、软件构造方法，甚至于软件过程、管理过程还能适应吗？这必然对软件工程提出新的挑战。未来面向服务的软件工程更是值得研究和期待。

1.7.3 从软件工程学到软件经济学

第 1.5.4 节提到的巴利·玻姆教授（Barry Boehm）是美国国家工程院院士，AIAA、IEEE、ACM 会员，现任美国南加州大学（University of Southern California，USC）TRW 软件工程教授，系统与软件工程中心（Center for Systems and Software Engineering，CSSE）的创始人。巴利·玻姆教授是软件业中最有影响的专家之一，对软件工程领域做出了诸多杰出贡献，其中包括软件成本估算的 COCOMO 模型（Constructive Cost Model）、软件过程中的螺旋模型（Spiral Model）、适用于软件管理和需求协商的 W 理论（win-win），以及编写了奠定软件成本估算领域基础的经典著作《软件工程经济学》。他开创并发展的 COCOMO 模型一直以来引领着软件成本估算技术的发展，在全世界范围内得到了非常广泛的产业应用。因此，他被学术和产业界同行尊誉为“软件工程经济学之父”。

如果说“软件工程学”关注的是流程，那么“软件经济学”则更加关注结果。越来越多的团队开始使用“软件交付”（Software Delivery）来代替“软件开发”（Software Development）。从开发到交付的微妙差异，象征着管理理念的重大变革，即从关注“项目过程”转变到关注“项目目标”。换言之，以“软件开发”为导向注重的是在开发过程中涉及的多种活动，而以“软件交付”为导向则侧重于开发过程的最终结果。基于结果的软件管理，最终目标是使用有限的资源达到更好的结果，经济因素已抢占技术的主导地位。

2009 年 8 月，IBM Rational 软件高峰论坛（IBM Rational Software Conference China 2009，RSC）在北京和上海召开，IBM 发布了名为“提升软件经济”（Improve Software Economics）的白皮书。IBM 在会上首次提出了“软件经济学”理论[10]，强调依赖于软件

的组织应该经济地衡量投入与产出，以此指导软件开发中各项工作与资源的配比，以获得最佳的 ROI（投资回报率）。“软件经济学”是“软件工程学”的重大进步，可能将会引发软件产业革命。“软件工程学”面向软件开发流程，聚焦于开发过程中所需要的各个活动。而“软件经济学”面向“软件交付”，聚焦于过程的结果。“软件经济学”关注软件开发中价值的判定、成本的权衡、人性因素、宏观经济趋势、技术趋势以及市场状况和时机。

COCOMO Ⅱ模型是抽象为复杂度、过程、团队、工具 4 个基本参数的软件经济学的简化模型。面向“软件经济学”就是用降低复杂度/规模、改进过程、改进团队效率和工具自动化方面来优化 4 个参数以有效降低软件成本。

1.7.4　从软件工程应用到教学

通过上面的介绍，了解到软件工程知识体系如此庞大而且还在不断扩张，反观软件工程教学的内容、方法、手段却远远落后于实际应用，这可否称其为“软件工程教学危机”呢？在构思和写作本书的同时期内，基于架构的开发技术已经相当成熟，当前业界在企业级软件开发中把架构作为主流技术。如采用 Java 技术的 J2EE 架构和微软的.NET 架构，还有适应敏捷开发的 Rails 架构，面向服务架构 SOA 等。架构本身提供的分层结构、设计模式、中间件容器、开发平台，使得开发者不必再为软件体系结构及应用结构的划分和设计困扰，不必再为组件的运行管理、并发访问、事务管理、数据安全等高层设计而浪费时间和精力，让软件开发者集中精力处理业务逻辑问题，使得企业级软件的开发效率和成功率大为提高。这种基于架构技术也就是组件技术的开发平台给软件开发带来革命性的变化，而流行在国内的“软件工程”类教材没能跟得上这个变化，在基于架构软件开发方面却依然沿用传统的面向对象方法，没有针对架构归纳出新的方法。按传统的面向对象方法不是利用架构提供给我们的优化方案简化工作，而是把这些先进的技术退化到传统的方法中去，弄巧成拙，弃简从繁，在分析和设计中由架构所带来的好处得不到体现。

在 1.7.1 节中已介绍了软件开发方法学始终滞后开发技术的现象，作为课堂教育及教材内容更是滞后于工程应用，我国的课堂教育滞后现象更为严重。比如 20 世纪 80 年代开始在国际上广泛流行面向对象开发，而我国当时软件工程的教育正津津乐道于面向结构化的开发方法。直到 20 世纪与 21 世纪之交时面向对象开发方法才被写进软件工程的教材并慢慢普及开来，而且一直延续到今天。而今天面临架构的成熟应用，讲授开发方法却依然沉浸在传统的面向对象开发方法上，它已远远不能胜任今天的面向架构开发技术。

本章小结

本章作为学习软件工程课程的引导，主要引导读者进行如下方面内容的学习。

（1）介绍了有关软件工程的一些基本概念，概念的理解顺序是：软件→软件危机→软件工程。

① 软件：通过介绍软件的概念，纠正由于以程序设计作为学习先导而建立起来的观念上的缺陷，不要认为软件就是程序，软件＝程序+数据+文档。

② 软件危机：通过介绍软件的发展过程阐述了软件危机产生的原因。找到病因也就找到了治疗的办法。重点了解软件危机产生的原因和表现。

③ 软件工程：是解决软件危机的正确途径。软件工程所涵盖的内容极其广泛，它确实是一个综合性的学科，仅用一段定义是不能表达完整的，甚至于任何一部书都不能描述全面。在 1.5 节中从 4 个角度对软件工程所涉及的内容进行了简单的描述，从中可以学习到软件工程定义、软件工程三要素、软件工程三维全景、软件工程基本原理和学习思维的转变。

（2）本章介绍软件工程的发展，目的是让读者了解软件行业的一个特殊现象，即开发方法滞后于技术的进步，或者说技术引领开发方法的进步，并且依然影响到今天。

① 通过介绍软件前三个阶段的发展来了解软件危机的产生和软件工程的诞生。

② 介绍软件发展的第 4 和第 5 阶段的目的是了解现在和即将应用的技术，看到开发方法学与技术方面存在的新差距，开阔视野，找出努力的方向。

③ 架构与 MVC 设计模式相结合进行软件开发是本书要讲述的核心问题，因此本章以介绍软件工程的发展为切入点，逐步引出架构技术。今天架构的广泛使用，导致软件开发发生很大变化，这种变化的趋势是开发越来越简单、越来越容易。反观开发方法并没有与其同步发展，尤其是教材与教学更是相形见绌。

（3）了解软件工程的演变发展也是要引导读者关注现代软件工程。现代软件工程的发展内容包括：

① 适合现代技术的开发方法学；

② 更注重管理，融合了配置管理、质量管理、过程改进等许多新的管理理念。

习 题

一、基本概念

1．软件就是程序吗？如何定义软件？

2．文档有何作用？程序代码是否属于文档？

3．什么是软件危机？什么原因导致了软件危机？

4．软件工程定义有很多说法，用自己的理解说明软件工程是做什么的。

5．软件工程三要素是什么？

6．软件工程的目标、过程和原则是什么？

7．软件工程原理有哪几条？

8．软件发展的第 4 阶段比第 3 阶段的显著变化和进步表现在哪些方面？第 5 阶段的显著特征是什么？

9．传统软件工程曾起到哪些推进作用？

10．现代软件工程与传统软件工程显著区别是什么？有哪些进步？

二、深入讨论

1．辨析：软件危机是过去发生的事情，现在有了软件工程，软件危机得到有效解决，现在和以后不会再出现软件危机了。

2．讨论：请总结一下你使用过什么类型的软件。本书讨论的软件工程倾向于什么类别的软件开发？

3．讨论：企业级软件开发有何特点？

4．讨论：计算机科学方面的基础课程与学习软件工程有何关系？有何不同？