第 1 章　软件工程概述

计算机学科是 20 世纪发展最快的新兴学科，在短暂的 50 年里，计算机已经渗透到社会的各个领域，有力地推动了整个社会信息化的发展。进入 21 世纪以来，经济全球化的趋势加速，现代科学技术突飞猛进，市场竞争更加剧烈，人类面临着巨大的社会经济变革。在这个复杂多变的时代，每个人、每个社会组织对信息资源的开发利用能力已成为其竞争力的主要标志之一。

随着技术进步，计算机性能在不断地提高，计算机的体积、功耗、价格却不断下降。今天的计算机在科学计算、数据处理、过程控制、计算机辅助系统、人工智能等领域得到广泛的应用。尤其计算机与全球 Internet 相连接，使今天的社会进入了以计算机为核心的信息社会。在信息社会中，信息的获取、处理、交流和决策都需要大量高质量的计算机软件，软件系统无所不在。这样就促使人们对计算机软件的品种、数量、功能、质量、成本和开发时间等提出越来越高的要求。

为了使世界上丰富的软件资源为人类共享，人们越来越重视软件、软件开发及运行环境的标准化。计算机的各类程序设计语言和多媒体人机交互工具已被越来越多的人所掌握，成为世界性的文化现象。

软件工程是在 20 世纪 60 年代末期提出的。这一概念的提出，其目的是倡导以工程的原理、原则和方法进行软件开发，以期解决当时出现的“软件危机”。

1.1　软件工程中的常见问题

本节讨论软件工程的常见问题，并希望根据国内外专家的观点，按照“FAQ（常见问题表）”给出简明的答案，然后逐次加以简述，深入的研究在各章节进行。软件工程中常见问题如表 1.1 所示。

表 1.1　软件工程中常见问题（一）

问题	答案
什么是软件	计算机程序和相关文档。软件产品可为特定客户或通用市场开发
什么是软件危机	“软件危机”一词在 IT 界广为流传，主要针对软件代价高和软件错误多现象
什么是软件工程	软件工程是关于软件生产的各个方面的工程学科
软件工程和计算机科学有何区别	计算机科学侧重理论和基础，而软件工程侧重于软件开发和交付的实际活动
软件工程和系统工程有何区别	系统工程侧重基于计算机系统开发的所有方面，包括硬件、软件和处理工程。软件工程只是它的一部分
什么是软件过程	以软件开发和进化为目的的一系列活动
什么是软件生命周期	软件生命全过程。分为可行性研究、需求分析、概要设计、详细设计、实现、组装测试、确认测试、使用和维护 8 个阶段
什么是软件过程模型	从特定角度提出的软件过程的简化表示形式

续表

问题	答案
什么是软件工程成本	软件开发成本约占 60%，测试成本占 40%。对于定制软件而言，进化成本常常高于开发成本
什么是软件工程方法	软件开发的结构化研究方法，包括系统模型、标记法、规则、设计忠告和过程指南
什么是优良软件特点	软件应具有用户所需的功能与性能，而且应该可维护、可靠、可用
软件工程面临的主要挑战是什么	要面临正在使用的旧系统、不断增长的多样性以及减少递交次数等问题的挑战

1.1.1 软件

软件包括了使计算机运行所需要的各种程序及其有关的文档资料。其中，程序是计算机任务的处理对象和处理规则的描述；文档是为了理解程序所需的阐述性资料。

计算机软件主要包括系统软件与应用软件两大类。

系统软件是生成、准备和执行其他程序所需要的一组文件和程序，如操作系统（包括 DOS、Windows、UNIX 等）……

应用软件是计算机用户为了解决某些具体问题而购买、开发或研制的各种程序或软件包，包括字处理软件 Word、WPS、各种电子设备的控制系统、特定的业务处理系统等……

这两类产品的一个重要区别在于：在系统软件中，软件设计由开发者自己完成，而应用软件通常是由用户提出要求，开发者按用户要求进行开发。

1.1.2 软件危机

进入 21 世纪，软件已经使我们比以往任何时候更快、更有效地完成任务。软件支撑了在医学、农业、交通和经济和其他许多领域的改革。软件使我们能做以前不可思议的事情，如显微外科手术、多媒体教育、机器人技术等。

然而，软件并非没有问题，事实上，软件的缺陷和开发无缺陷软件的困难是科学文献和日常工作中经常讨论的问题。从 20 世纪 60 至 70 年代，“软件危机”一词在计算机界广为流传，其主要针对普遍存在的软件代价高和软件错误多的现象。

1. 软件代价高

计算机产业已被我们普遍认为是国民经济的一个重要组成部分。但很少有人会意识到计算机系统的耗费的巨大。在美国政府 1980 年的财政年度中，计算机系统方面竟耗费了大约 570 亿美元，而其中的 320 亿美元（占总数的 56%）是用于计算机软件方面的。人们逐渐开始意识到在开发一个新型计算机系统或修改一个现有系统的过程中，最大部分的资金是用在系统软件开发方面。图 1.1 表明了计算机系统硬件/软件成本变化趋势，工业界为维护软件支付的费用占全部硬件和软件费用的 40%～75%；许多重要的软件开发项目，在耗费了大量的人力与财力之后，由于离预定目标甚远，宣告失败。

2. 软件产品缺陷和故障

IEEE 已经提出了描述软件产品中“错误”的标准术语，解释了缺陷、错误和故障之间的关系。

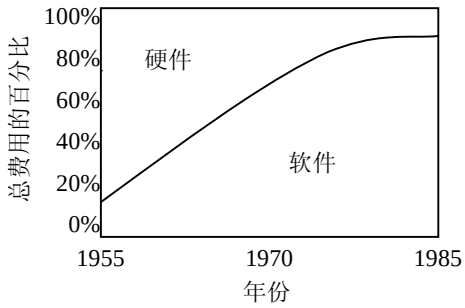

图 1.1　计算机系统硬件/软件成本变化趋势

通常，我们在软件中谈到的“错误”（Bug）有很多意思。错误可能是需求说明的错误、代码中的语法错误或引起系统崩溃的原因（该原因目前未知）。

缺陷（Fault）是人进行软件开发活动中人为出错（称为错误，Error）造成的。例如，一个设计者可能误解需求，并得出了与需求分析员和用户的实际需求不符的设计。这个设计缺陷是一种错误的代码，它能导致其他缺陷，如不正确的代码或用户手册中不正确的描述等。因此，单个错误可能产生多个缺陷，而且任何开发或维护过程中都可能存在缺陷。

故障（或称失效，Failure）是相对于系统指定行为的偏离。它可能会在系统交付前后、测试过程或在操作和维护的过程中被发现。既然需求文档可能包含错误，因此故障意味着系统即使按说明执行，也可能没有按照需要来执行。

因此，缺陷是系统的内部表现，而故障是外部表现：即用户所看到的问题。

下面来看几个故障的例子并了解出错原因。

20 世纪 80 年代早期，美国国内税收服务处委托 Sperry 公司建立一套联邦税收表格自动处理系统。根据华盛顿邮报的报道，“系统被证明不适合当前的工作量，花费几乎是预算的两倍，必须立即更换”（Sawyerl985）。1985 年，还需要再加 9 千万美元来改进 Sperry 公司最初价值 1.03 亿的设备。另外，因为出现的问题妨碍了 IRS 按时返还纳税者的税款，IRS 被迫偿付 4.02 千万美元的利息以及 2.23 千万美元的工资给加班职员。1996 年，情况并没有得到改善。洛杉矶时报在 3 月 29 日报道说，仍然没有更新 IRS 计算机系统的成熟计划，目前只有一个 6000 页的技术文档。美国国内舆论把这个项目称为“40 亿美元的彻底失败，原因是没有充分计划就错误行事”。

尽管许多软件供应商致力于设计零缺陷的软件，事实上大多数软件产品都做不到。许多软件工程师认为反弹道导弹系统至少需要 1 千万行代码；有些人估计竟高达 1 亿行。事实是，支持美国航天飞机的软件只包含 300 万行代码，包括控制发射和飞行的地面控制计算机；1985 年，航天飞机上只有 10 万行代码。因此，一个反导弹软件系统将需要数量巨大的代码测试。实际上，可靠性约束问题是无法测试的。一般地，我们说某些事情是安全攸关则其可靠性应当至少是 10^{-9}。这意味着系统运行 10^9 小时其失效不能超过一次。要观察这个数量级的可靠性，就不得不使这个系统运行至少 10^9 小时验证它没有失效，但是 10^9 小时超过 114000 年——作为一个测试周期来说它实在是太长了。

如果不能适当地设计软件或进行编码，原本有用的技术可能会致命。例如，当 Themc-25（一种射线疗法和 X 射线机器）发生故障并使几个病人死亡时，医学界变得惊恐万状。软件设计者没有估计到会有不按标准使用几个方向键的情况；结果，当需要低剂量的射线束时，软

件却保持高剂量的设置并发出了极为集中的射线束。

西方计算机科学家把软件开发和维护过程中遇到的一系列严重问题统称为“软件危机”，其表现为：

（1）不能正确地估计软件开发成本和进度，致使实际开发成本往往高出预算很多。

（2）软件产品不可靠，满足不了用户的需求，甚至无法使用。

（3）交付使用的软件不易演化，以致于人们不得不重复开发类似的软件。

（4）软件生产率低下，远远满足不了社会发展的需求。

（5）软件缺乏适当的文档资料。

以上列举的仅仅是软件危机的一些明显的表现，在实际应用中与软件产品缺陷和故障有关的问题远不止这些。

3. 克服危机的途径

（1）应该加强软件开发过程的管理，做到组织有序、各类人员协同配合，共同保证工程项目完成，避免软件开发过程中个人单干的现象。

（2）推广使用开发软件的成功技术与方法，并且不断探索更好的技术与方法；消除一些在计算机系统早期发展阶段形成的一些错误概念和做法。在软件设计活动的整个过程中必须考虑对系统意料之外的使用。拓展你的想象力去想象系统能如何被滥用（与想象如何正确使用一样），还可以假定系统将被滥用并设计软件处理这种情况。

（3）开发和使用好的软件工具，支持软件开发的全过程，即建立软件工程支持环境。

总之，为了解决软件危机，要从技术、管理两个方面入手，引入“软件工程”的概念，从而解决软件开发过程中的技术和管理问题。

1.2 软件工程

1968 年和 1969 年北大西洋公约组织成员国软件工作者两次召开会议（NATO 会议），讨论摆脱软件危机的办法，提出了软件工程的概念，试图建立并使用正确的工程方法开发出成本低、可靠性好并能高效运转的软件，从而解决或缓解软件危机。

1.2.1 软件工程的定义与原理

根据国际电子与电气工程师协会（IEEE）给出的定义，软件工程是：

（1）将系统化、严格约束、可量化的方法应用于软件的开发、运行和维护，即将工程化应用于软件。

（2）将工程化应用于软件方法的研究。

在软件工程的定义中有两个关键概念：

第一，工程学科。干什么事情都离不开工程人员，他们既能恰当地应用理论、方法和工具，又能有选择地利用它们，即使在没有可用的理论和方法的情况下，也力求找出解决问题的方法。同时他们也认识到必须在行政或财政状况所允许的限度内工作，即要在此限度内寻找解决办法。

第二，软件生产的各个方面。软件工程不仅涉及软件开发的技术过程，也涉及诸如软件项目管理、支持软件生产的工具、方法和理论的开发等活动。

总之，软件工程人员运用的是系统的、有组织的工作方法，这种方法对于制作高质量的软件是最有成效的。而所谓工程就是为某种情况选择最恰当的解决办法，而更具创造力的、非正规的开发方法在某些情况下可能是有效的。非正规开发尤其适用于基于 Web 的电子商务系统，该系统需要软件和图形设计技巧的相互融合。

著名的软件工程专家B.W.Boehm于1983年综合研究了软件工程的专家与学者们的意见并总结开发软件的经验，提出了软件工程的 7 条基本原理。这 7 条基本原理被认为是迄今为止软件工程准则的完美结合。

（1）用分阶段的生命周期计划严格管理。把软件开发与维护的过程称为软件生命周期，B.W.Boehm 认为软件整个生命周期应该分成 6 个步骤，即制定计划、需求分析、设计、程序编码、测试及运行维护。

（2）坚持进行阶段评审。在每个阶段都进行严格的评审，及早发现软件开发过程中的错误，可以减少错误造成的损失，尤其是设计阶段的错误占软件错误的 63%。

（3）实行严格的产品控制。依靠科学的产品控制技术来顺应用户提出的改变需求的要求，其中关键技术是实现基准配置管理，一切修改，特别是涉及对基准配置的修改，必须经过批准才能实施。

（4）采用现代程序设计技术。20 世纪 60 年代末到 80 年代初，软件系统的规模、复杂性以及在关键领域的广泛应用，促进了软件开发过程的管理及工程化开发。围绕软件项目，开展了有关开发模型、支持工具以及开发方法的研究。这一时期的主要特征可概括为：前期主要研究系统实现技术，后期则开始强调管理及软件质量。

（5）结果应能清楚地审查。完成软件开发项目的总体目标，在给定成本、目标进度的前提下，规定开发组织的责任和产品标准，从而保证结果可以清楚地审查。

（6）开发小组的人员应该少而精。开发小组的人员素质好可降低软件中的错误，开发小组人员数目的减少，其相应的开销也随之减少。

（7）承认不断改进软件工程实践的必要性。不仅要积极主动采纳新的软件技术，而且要不断总结经验，以供今后软件开发借鉴。

1.2.2　软件工程与计算机科学

从本质上讲，计算机科学研究的是构成计算机和软件系统基础的有关理论和方法，而软件工程则研究软件制作中的实际问题。正如电子工程师必须具有一定的物理学知识一样，软件工程人员同样必须具有一定的计算机科学知识。

理论上，所有软件工程都应该以计算机科学理论作为坚实的基础，但实际情况并非如此。软件工程人员常常必须要用特定的方法去开发软件。对于实际、复杂的问题，计算机科学的经典理论不可能总是适用的，这时就需要应用软件工程的方法来解决。主要体现在以下 5 个方面。

1. 建设环境的复杂性

现代软件应用对象一般来说结构复杂，软件工程建设通常要涉及到用户组织内部与外部环境。软件工作者必须十分重视、深刻理解系统面临的内、外环境及发展趋势。系统的目标、规模、功能和实施步骤必须与用户当前的发展水平与能力相适应。

2. 用户需求的多样性

软件开发失败最主要的原因是：用户对软件需求的描述不精确，可能有遗漏、有二义性、

有错误，甚至在软件开发过程中，用户还不断提出修改软件功能、界面、支撑环境等方面的要求；急于求成，软件开发人员对用户需求的理解与用户的本来愿望不一致就着手编写程序，最终导致软件在没有投入使用前就遭到否定。

3. 建设内容的复杂性

软件不同于硬件，它是计算机的逻辑部件而不是物理部件，在写程序代码并上机运行之前，软件开发过程的进展情况较难衡量，软件开发的质量也较难评价，因此管理和控制软件开发过程相当困难。

4. 技术手段的复杂性

此外，软件运行中发现的错误，很可能是在开发时期引入而在测试阶段没能检测出来的故障，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，软件设计、实施、维护技术手段的复杂性，也是造成软件危机的一个重要原因。

5. 所需资源的密集性

软件系统是资金、劳动、智力、知识密集型大型项目，开发需要组织一定的人力共同完成，各类的信息交流不及时、不正确、甚至产生误解，这也是产生软件危机的主要原因。

1.2.3 软件工程目标

软件工程的目标可概括为：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。

将软件工程的首要目标、软件工程的各分阶段目标概括如表 1.2 所示。

表 1.2 软件工程的目标

首要目标	各阶段目标	项目追求的和应该达到的标准
软件定义可靠性	可适应性（adaptability）	软件在不同的系统约束条件下，使用户需求得到满足的难易程度
	有效性（efficiency）	软件系统在某个给定时刻根据规范程序正确运行的比率
软件开发可靠性	可理解性（understandability）	系统具有清晰的结构，能直接反映问题的需求
	可互操作性（interoperability）	多个软件元素相互通信并协同完成任务
	可重用性（reusebility）	软件在可以适应多种场合应用的程度
软件使用与维护可靠性	可修改性（modifiability）	容许对系统修改而不增加复杂度
	可追踪性（traceability）	根据软件需求对软件设计、程序进行正向追踪，或根据程序、软件设计对软件需求进行逆向追踪的能力
	可维护性（maintainability）	软件产品交付用户使用后，能够对它进行修改，以便改正潜伏的错误，改进性能和其他属性
	可移植性（portability）	软件从一个计算机系统或环境搬到另一个计算机系统或环境的难易程度

追求软件产品的这些目标有助于提高软件产品的质量和开发效率，减少维护的困难。实现这些目标就能保证软件的可靠性。

应该特别指出；“可靠性”这个目标在软件工程中有着重要的意义。广义上讲，它涉及到产品设计的一系列问题，从而使产品能在相当长的期间内稳定工作。狭义上讲，可靠性是软件

成功运行的概率度量，可靠性分析和可靠性测试就可作为衡量软件质量和其他开发过程的最重要的方法之一。

1.2.4　软件工程面临的挑战

软件工程在 21 世纪面临三大挑战。

1. 遗留系统的挑战

我们现在使用的大多数大型软件系统都是许多年前开发的。时至今日依然担负着重任。对遗留系统而言面临的问题是：维护和更新这些软件，既要避免过多的支出，又要不断地满足基本的业务服务。

2. 多样性的挑战

多样性带来的挑战是：必须开发新技术，制作可靠的软件，从而满足多样性的要求。

3. 交付上的挑战

许多传统的软件工程技术需要耗费大量的时间，用于提高软件质量。而今天的软件制作必须响应快、更换迅速，支持软件也必须同样快地进行更换。交付上的挑战是：在不损及系统质量的前提下，缩短大型、复杂系统的移交时间。市场压力促使软件开发人员尽快交付产品，几乎没有时间进行完整的测试。为了应对这些挑战，我们需要有新的工具和技术以及融合使用现有软件工程方法的创新举措。

1.3　软件过程和软件生命周期

1.3.1　软件过程

本节讨论新形势下软件工程的开发过程，软件工程建设的复杂性和系统科学方法的主要原则，以及软件生命周期和软件开发模型等知识，通过学习这些内容，为后续章节的学习打好概念与方法论基础。

上述的软件开发目标适用于所有的软件系统开发。为了达到这些目标，在软件开发过程中围绕工程设计、工程支持以及工程管理，提出了软件工程的框架及软件工程的 4 条基本原则，如图 1.2 所示。

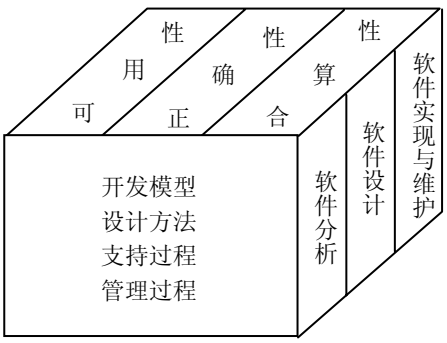

图 1.2　软件工程的框架

第一条原则是选取适宜的开发模型。该原则与系统设计有关。在系统设计中，软件需求、硬件需求以及其他因素之间是相互制约、相互影响的，经常需要权衡。因此，必须认识需求定义的易变性，采用适宜的开发模型予以控制，以保证软件产品满足用户的要求。

第二条原则是采用合适的设计方法。在软件设计中，通常要考虑软件的模块化、抽象与信息隐蔽、局部化、一致性以及适应性等特征。合适的设计方法有助于这些特征的实现，以达到软件工程的目标。

第三条原则是提供高质量的工程支持。“工欲善其事，必先利其器”。在软件工程中，软件工具与环境对软件过程的支持颇为重要。软件工程项目的质量与开销直接取决于对软件工程所提供的支撑质量和效用。

第四条原则是重视开发过程的管理。软件工程的管理，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，仅当软件过程予以有效管理时才能实现有效的软件工程。

综上所述，软件工程框架告诉我们，软件工程目标是可用性、正确性和经济性；实施某个软件工程要选取适宜的开发模型，要采用合适的设计方法，要提供高质量的工程支撑，要实行开发过程的有效管理；软件工程活动主要包括需求、设计、实现、确认和支持等活动，每一活动可根据特定的软件工程，采用合适的开发模型、设计方法、支持过程以及过程管理。根据软件工程这一框架，软件工程学科的研究内容主要包括：软件开发模型、软件开发方法、软件过程、软件工具、软件开发环境、计算机辅助软件工程（CASE）以及软件经济学等，根据需要这些内容将在各章分别阐述。

1.3.2 软件生命周期

软件生命周期（Software Life Cycle）表明一个计算机软件从功能确定、设计，到开发成功投入使用，并在使用中不断地修改、增补和完善，直至被新的需要所替代而停止该软件的使用的全过程。根据软件所处的状态、特征以及软件开发活动的目的、任务，软件生命周期可以划分为若干个时期，而每一个时期又进一步划分为若干个阶段。

我国国家标准《计算机软件开发规范》（GB8566-88）把软件生命周期划分为可行性研究、需求分析、概要设计、详细设计、实现、组装测试、确认测试、使用和维护 8 个阶段。通常我们把可行性研究、需求分析称为软件定义时期，把概要设计、详细设计、实现、组装测试、确认测试称为软件开发时期，而把使用与维护阶段称为软件运行维护时期。

软件生命周期各个时期及阶段关系如图 1.3 所示。

1. 软件需求定义

软件定义又称为需求过程，基本任务是确定软件系统的工程需求，也就是确定用户要求开发的系统“做什么？”。在此过程中开发人员开发系统模型，此过程包括：

（1）可行性研究。确定待开发软件系统的总目标，给出其功能、性能、可靠性、接口等要求。在其基础上对项目从技术、经济和社会等多方面进行可行性调研和论证，即该项目是否值得去开发，是否存在可行的解决办法，并提交可行性论证报告和项目开发报告。

（2）需求分析和定义。系统员对用户的需求进行分析并给出确切的定义，软件人员和用户讨论共同决定哪些需求是可满足的并对其加以确切的描述。然后编写出软件需求说明书及系统用户手册，交管理机构评审。

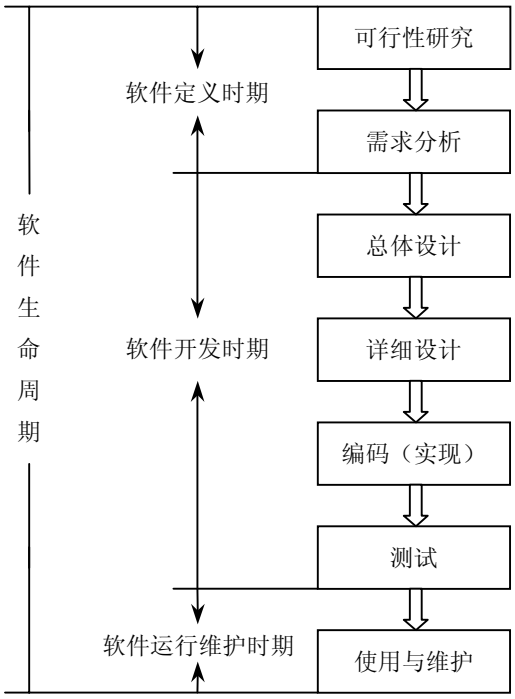

图 1.3　软件生命周期各个时期及阶段的关系

2. 软件开发

（1）总体设计。又称概要设计，是软件工程的技术核心。在此阶段必须回答的关键问题是系统“怎么做？”，软件设计人员从已获取的各项需求，得出系统的功能模块，确定各模块的输入、输出以及相互联系。

（2）详细设计。详细设计阶段的任务是确定系统机构中的每一个模块的算法、数据结构以及程序模块间的接口信息等内部细节，设计出程序的详细规格说明。

（3）编码。根据详细规格设计说明用某种选定的程序设计语言把详细设计的结果转化为机器可运行的源程序模块。这是一个编写程序和调试的过程。

（4）软件测试。软件测试是保证软件质量和可靠性的重要手段，其主要方式是在设计测试用例的基础上检测软件的各个组成部分。首先进行单元测试，检查各模块功能和结构上存在的问题并加以纠正；其次进行组装测试，将测试过的模块按照一定的次序组装起来；最后按规定的各项要求，逐项进行有效性测试，决定已开发的软件是否合格。

3. 软件运行与维护

软件使用与维护是指实现过程，在此过程中开发人员实现每个组件，包括：

（1）软件运行。将软件安装在用户确定的运行环境中使用。

（2）软件维护。软件投入使用后，还要进行维护工作，维护的主要任务是通过各种必要的维护活动使系统持久地满足用户的需求。

（3）退役。软件的退役是指软件一旦完成使命，或者由于一个新的软件生命周期的开始，就要终止对软件的支持，即停用。

综上所述，在划分软件生命周期的阶段时应该遵循一条基本的原则就是使各阶段的任务之间尽可能相对独立，同一阶段各项任务的性质尽可能相同，从而降低每个阶段任务的复杂程度，简化不同阶段之间的联系，有利于软件开发工程的组织管理。在软件生命周期的每个阶段都采用了科学的管理和良好的方法与技术，同时在每个阶段结束之前都应该从技术和管理的角度进行严格的审查，合格之后才开始下一阶段的工作，这就使软件开发工作具有一定的可控性。从而保证了软件质量，特别提高了软件的可维护性。

1.4 软件开发模型

软件开发模型（软件生存周期模型）是从软件项目需求定义直至软件经使用后废弃为止，跨越整个生存期的系统开发、运作和维护所实施的全部过程、活动和任务的结构框架。软件开发模型大体上可分如下几种类型：瀑布模型、渐进式模型、原型模型、螺旋模型、喷泉模型等。在实际开发时，可以把几种模型结合起来使用以便充分利用各种模型的优点。

1.4.1 瀑布模型

瀑布模型（Waterfall Model）又称为软件生命周期瀑布模型，是传统的开发模型，是由W.Royce 于 1970 年提出的。该模型将软件生存周期的各项活动规定为依序连接的若干阶段工作，从问题定义开始逐一按生命周期各阶段顺序进行，直至得到用户确认。各阶段的工作自顶向下。瀑布模型软件过程的基本框架如图 1.4 所示。

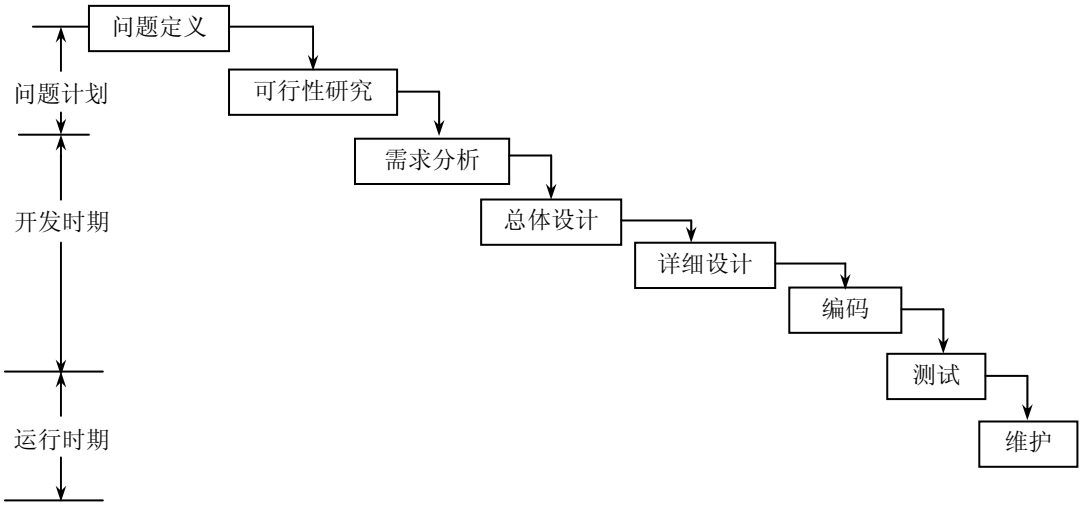

图 1.4 瀑布模型

当采用瀑布模型进行开发组织时，应制定软件开发规范或开发标准。其中要明确规定各个开发阶段应交付的产品，这就为严格控制软件开发项目的进度，最终按时交付产品以及保证软件产品质量创造了有利条件。

为了保障软件开发的正确性，每一阶段任务完成后，都必须对它的阶段性产品进行评审，确认之后再转入下一阶段的工作。因此，它是一种以文档作为驱动的模式。例如在评审过程中，

如果发现错误和疏漏，应该反馈到前面的有关阶段修正错误、弥补疏漏，然后再重复前面的工作，直至某一阶段通过评审后再进入下一阶段。这种形式的瀑布模型是带有反馈的瀑布模型，如图 1.5 所示。

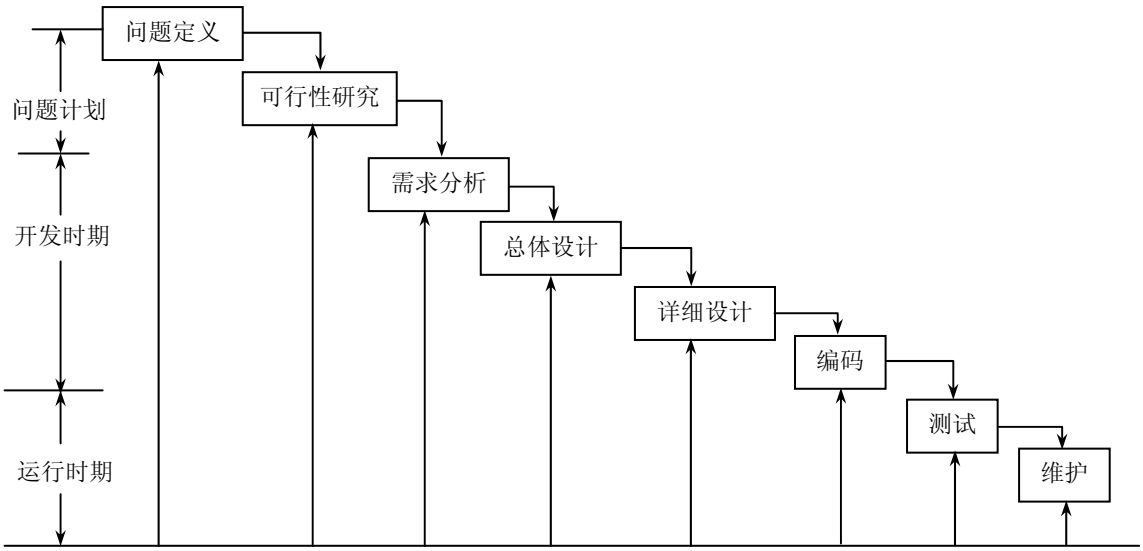

图 1.5　带有反馈的瀑布模型

瀑布模型 20 多年来之所以广泛流行，是因为它在支持结构化软件开发、控制软件开发的复杂性、促进软件开发工程化等方面具有显著作用。它提供了软件开发的基本框架，有利于大型软件开发过程中人员的组织、管理，有利于软件开发方法和工具的研究与使用，提高了大型软件项目开发的质量和效率。与此同时，瀑布模型在大量软件开发实践中也逐渐暴露出它的缺点，其中最为突出的缺点是该模型缺乏灵活性，无法通过开发活动澄清本来不够确切的软件需求，而这些问题可能导致开发出的软件并不是用户真正需要的软件，反而要进行返工或不得不在维护中纠正需求的偏差，为此付出高昂的代价，为软件开发带来不必要的损失。并且，随着软件开发项目规模的日益庞大，该模型的不足所引发的问题显得更加严重。

1.4.2　演化模型

演化模型（Evolutional Model）主要针对用户事先不能完整定义需求的软件开发项目，在逐个项目实施过程中需要用户与软件开发人员的密切配合。用户可以先提供待开发系统的核心需求，当看到核心需求实现后，有效地提出反馈，以支持系统的最终设计和实现。

软件开发人员根据用户的需求，首先开发核心系统。当该核心系统投入试运行后，用户试用之，完成他们的工作，并提出精练系统、增强系统能力的需求。软件开发人员根据用户的反馈，实施开发的迭代过程。每一迭代过程均由需求、设计、编码、测试、集成等阶段组成，为整个系统增加一个可定义的、可管理的子集。具体过程如图 1.6 所示。

如果在一次迭代中，有的需求不能满足用户的要求，可在下一次迭代中予以修正。演化模型在一定程度上克服了瀑布模型的缺点，减少了软件开发活动的盲目性，但整个软件开发活

动持续进行，应该承认不断进行软件产品修改、维护的必要性，因此软件的维护、修改的成本较高。

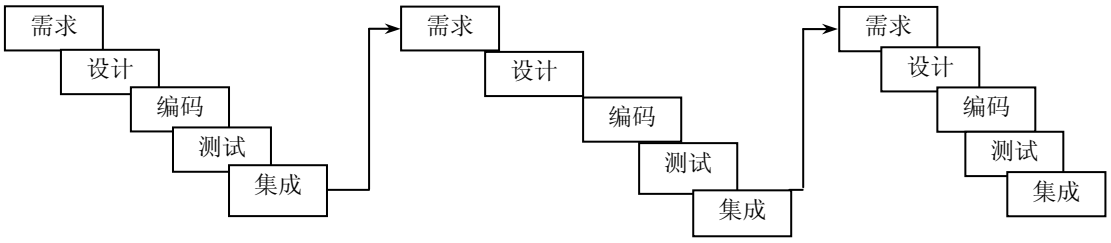

图 1.6　演化模型

1.4.3　原型模型

原型模型（Prototype Model）是软件开发人员针对软件开发初期在确定软件系统需求方面存在的困难，借鉴建筑师在设计和建造原型方面的经验，根据客户提出的软件基本要求，快速地开发一个原型。它向客户展示了待开发软件系统的全部或部分功能和性能，在征求客户对原型意见的过程后，进一步修改、完善原型，并在此基础上迭代，直至软件开发人员和用户确认软件系统的需求并达到一致的理解。

原型开发模型突出一个“快”字，即快速以感性的“样机”形式，在需求方面很快与用户达成共识，这不仅大大缩短了开发人员与用户反复讨论确定用户需求所占用的时间，而且能够很快获得一个全面、完整、准确的用户需求。利用原型定义和确认软件需求之后，就可以对软件系统进行设计、编码、测试和维护。原型开发模型如图 1.7 所示。

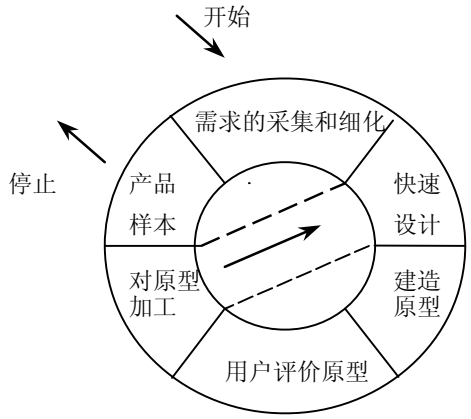

图 1.7　原型开发模型

下面分别介绍快速开发原型的过程。

1. 快速分析

在分析者和用户的紧密配合下，快速确定软件系统的基本要求。根据原型所要体现的特性（界面形式、处理功能、总体结构、模拟性能等），描述基本规格说明，以满足开发原型的需要。快速分析的关键是要注意选取分析和描述的内容，围绕使用原型的目标，集中力量，确定局部的需求说明，从而尽快开始构造原型。

2. 构造原型

在快速分析的基础上，根据基本规格说明，尽快实现一个可运行的系统。如软件的可见部分，包括数据的输入方式、人机界面、数据原型开发模型的输出格式等。由于原型是客户和软件开发人员共同设计和评审的，因此利用原型能统一客户和软件开发人员对软件项目需求的理解，有助于需求模型的定义和确认。

初始原型的质量对于原型生存期的后续步骤的成败是至关重要的。如果它有明显的缺陷，会带给用户一种不好的感觉；如果为追求完整而做得太大，就不容易修改，会增加修改的工作

量。因此，应当有一个好的初始原型。

3. 运行和评价原型

这是频繁通信、发现问题、消除误解的重要阶段。其目的是验证原型的正确程度，进而开发新的并修改原有的需求。它必须通过所有相关人员的检查、评价和测试。

1.4.4　螺旋模型

螺旋模型（Spiral Model）是在 1988 年由 TRW 的 B.Boehm 提出的。

螺旋模型是生存周期模型与原型模型的结合，它不仅汇集了两个模型的优点，而且还增加了新的成分——风险分析，使该模型更趋于优化。螺旋模型结构如图 1.8 所示，它由 4 个部分组成：需求定义、风险分析、工程实现、评审。螺旋模型是由上面 4 个部分组成的迭代模型。螺旋模型的每一周期都包括需求定义、风险分析、工程实现和评审 4 个阶段。

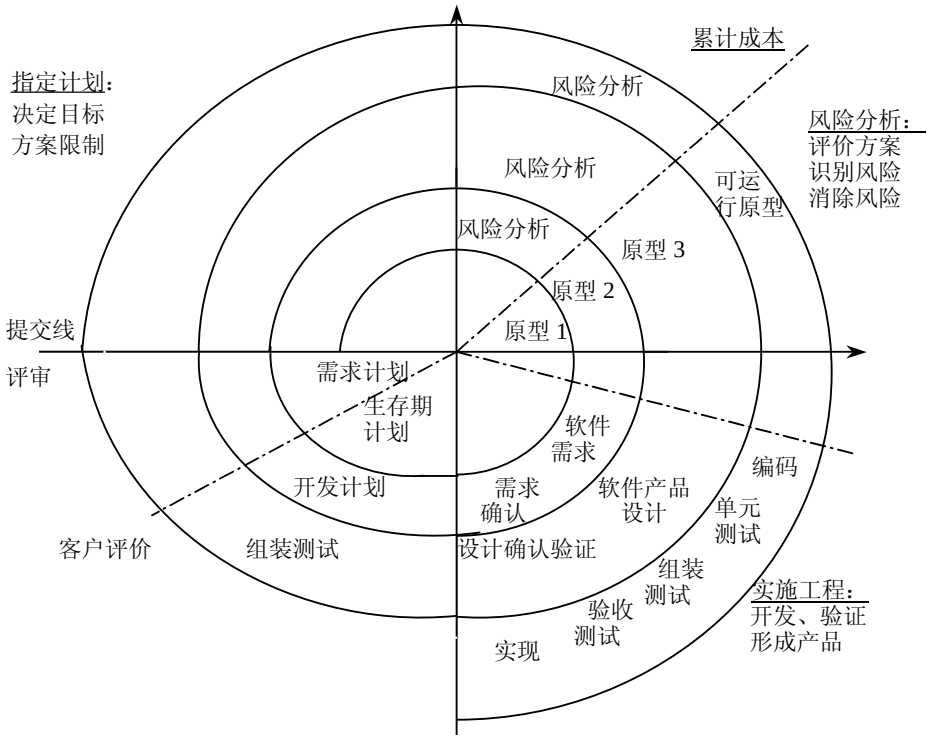

图 1.8　螺旋模型的结构

开发过程每迭代一次，螺旋线就增加一周，软件开发又前进一个层次，系统又生成一个新版本，而软件开发的时间和成本又有了新的投入，直到最后得到一个客户满意的软件版本。理论上，迭代过程可以无休止地进行下去，但在实践中，迭代结果必须尽快收敛到客户允许或可接受的目标范围内。只有降低迭代次数，减少每次迭代的工作量，才能降低软件开发的时间和成本，得到客户支持，软件系统开发才不至于中途夭折。

在需求分析阶段使用螺旋模型方法，必须从系统结构、逻辑结构、用户特征、应用约束、项目管理和项目环境等多方面来考虑，以决定是否采用螺旋模型方法。提交一个初始版本所需要的时间因问题的规模、复杂性、完整程度的不同而不同。3～6 周提交一个系统的初始版本

应是可能的，最大限度不能超过两个月。两个月后提交的应是系统而不是一个初始版本。

螺旋模型支持大型软件的开发，适合面向规格说明、面向过程、面向对象以及几种组合的软件开发方法。螺旋模型有着较好的应用前景。

1.4.5 喷泉模型及面向对象的 开发过程

喷泉模型（Water Fountain Model）是近几年提出来的软件生存期模型。它是以面向对象的软件开发方法为基础，以用户需求为动力，以对象为驱动的模型。“喷泉”一词本身体现了迭代和无间隙特性。喷泉模型如图 1.9 所示。

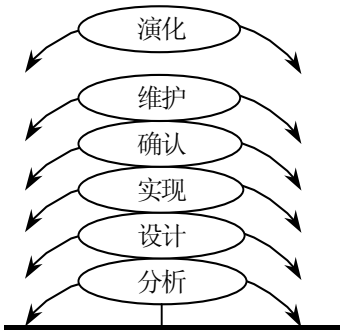

图 1.9 喷泉模型

迭代是指系统中某个部分常常重复工作多次，如软件刻画活动需要多次重复。例如，在编码之前（实践之后）再次进行分析和设计，其间添加有关功能，使系统得以演化。

同时，上述模型还表明软件开发活动之间没有明显的间隙，所谓无间隙是指在开发活动中，即分析、设计和编码之间不存在明显的边界。例如在分析和设计之间没有明显的界限。喷泉模型主要用于支持面向对象开发过程。

面向对象软件开发过程的特点是：由于对象概念的引入，使分析、设计、实现之间的表达没有明显间隙。面向对象软件开发过程（软件生命周期）表示如图 1.10 所示。

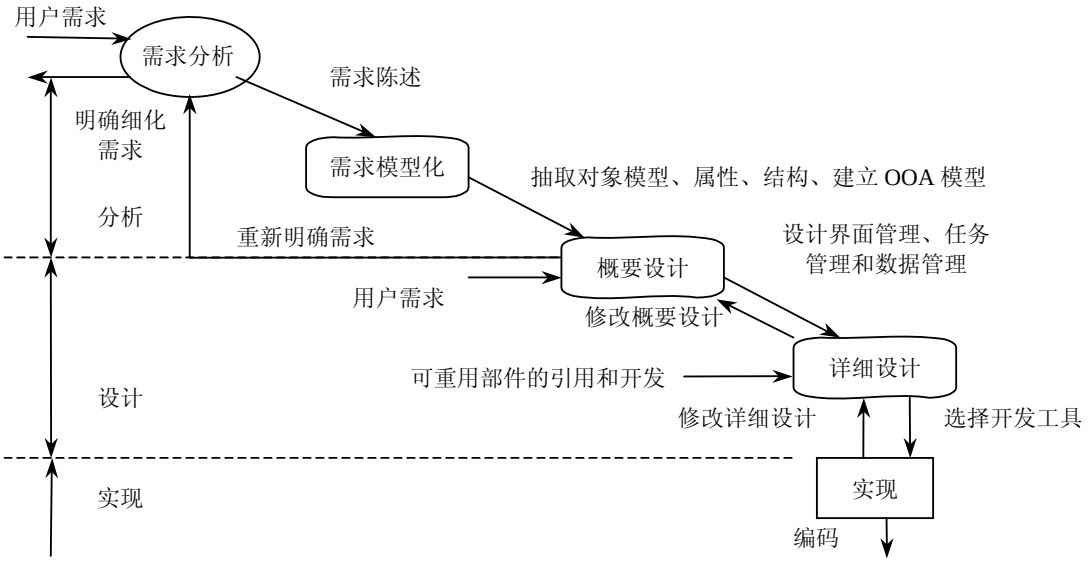

图 1.10 面向对象软件开发过程（软件生命周期）

1.4.6 基于四代技术的模型

4GL（四代语言）是 R.Ross 于 1981 年提出的，4GL 是面向结果的非过程式语言。以 4GL 为核心的软件开发技术称为四代技术。其软件开发模型如图 1.11 所示。

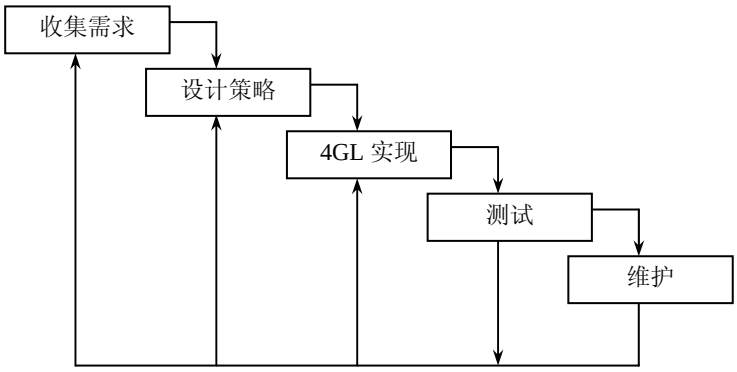

图 1.11　基于四代技术的模型

四代技术（4GL）是以第四代语言为核心的软件开发技术。基于四代技术的模型是指用 4GL 工具将开发者做出的软件规格说明自动转换成程序代码。这就大大缩短了设计和编码的工作。目前，支持 4GL 的软件开发工具有屏幕生成器、报表生成器、数据库查询语言、代码生成系统等。

1.4.7　智能模型

智能模型（Intelligence Model）也称基于知识的软件开发模型，它综合了上述若干模型，并与支持分析、测试、维护等应用领域的基于规则的专家系统相结合，采用归约和推理机制，帮助软件人员完成开发工作，并使维护在系统规格说明一级进行。为此，建立了知识库，将模型、软件工程知识与特定领域的知识分别存入知识库。以软件工程为基础的生成规则构成的专家系统与含有应用领域知识规则的其他专家系统相结合，构成了这一应用软件领域的开发系统。

1.5　职业道德

在软件工程的有些方面、某些行为没有法律加以规范，只能靠职业道德来约束，在这一方面，职业协会和机构肩负重任。国际电子与电气工程师协会（IEEE）提出了一个关于软件工程技术人员职业道德和职业行为的准则，其中谈到：

计算机在商业、工业、政府、医药、教育、娱乐和整个社会中的核心作用日渐突出。软件工程人员直接参与或讲授软件系统的分析、描述、设计、开发、认证、维护和测试。基于他们在软件系统开发中的地位，软件工程人员可能将事情做好也可能做坏，还可能让他人或影响他人将事情做好或做坏。为了最大限度地保证他们的工作是有益的，软件工程人员必须保证使软件工程业成为对社会有益的、受人尊敬的行业。软件工程人员应当做出承诺，使软件的分析、描述、设计、开发、测试和维护等工作对社会有益且受人尊重。基于对公众健康、安全和福利的考虑，软件工程人员应当遵守以下 8 条原则：

（1）公众感。软件工程人员应始终与公众利益保持一致。

（2）客户和雇主。软件工程人员应当在与公众利益保持一致的前提下，满足客户和雇主的最大利益。

（3）产品。软件工程人员应当保证他们的产品及其相关附件达到尽可能高的行业标准。

（4）判断力。软件工程人员应当具有公正和独立的职业判断力。

（5）管理。软件工程管理者和领导者应当拥护并倡导合乎道德的有关软件开发和维护的管理方法。

（6）职业感。软件工程人员应当弘扬职业正义感和荣誉感，尊重社会公正利益。

（7）同事。软件工程人员应当公平地对待和协助每一位同事。

（8）自己。软件工程人员应当毕生学习专业知识，倡导合乎职业道德的职业活动方式。

1.6 本章小结

本章主要介绍软件危机与软件工程的基本概念以及软件从业人员的职业道德。

为了解决软件危机，要从技术、管理两个方面入手，引入“软件工程”的概念，就是为了解决软件开发过程中的技术和管理问题。

软件产品由开发的程序及相关文档构成。软件产品的基本属性是可维护性、可依赖性、有效性、可用性。

软件工程的目标是：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。

软件生命周期实质上是大型系统开发过程中各个项目阶段的一种表示方法，上述基本的过程活动进一步展开，可以得到软件生命周期的 6 个步骤，即制定计划、需求分析、设计、程序编码、测试及运行维护。

软件生命周期模型是从软件项目需求定义直至软件经使用后废弃为止，跨越整个生命周期的系统开发、运作和维护所实施的全部过程、活动和任务的结构框架。

软件工程人员对软件工程这一职业和社会负有责任，不应该只关心技术问题。

习题

1. 试论述“软件危机”产生的原因和解决方法。

2. 软件危机最严重的征兆也许是低质量软件的开发。根据你自己的经验，如何区分“好的”（高质量的）软件和“差的”（低质量的）软件？

3. 有人说：软件开发时，一个错误发现得越晚，为改正它所付出的代价就越大。这个提法对否？请解释你的回答。

4. 软件工程的目标是什么？

5. 软件工程学的基本原则有哪些？为什么？

6. 说明“软件生命周期”的概念。

7. 论述瀑布模型软件开发方法的基本过程。

8. 举例说明哪些项目的开发适于采用原型与螺旋模型，哪些不适于采用这两种模型。

9. 总结各种开发模型的特点。

10. 软件工程人员应当遵守哪些基本原则？