第 1 章　软件工程概述

1.1　软件的概念、特点和分类

1.1.1　软件的概念

软件是软件工程学中的一个重要概念，任何一种计算机系统都包含硬件（Hardware）和软件（Software）两大部分。Software 一词，有人译为“软制品”，也有人译为“软体”，现在统一称为“软件”。许多人认为软件就是程序，那么，软件究竟是不是程序呢？

软件的定义如下：软件是计算机系统中与硬件相互依存的另一部分，它是包括程序、数据及其相关文档的完整集合。其中，程序是按事先设计的功能和性能要求编写的指令序列；数据是使程序能正常操纵信息的数据结构；文档是与程序开发、维护和使用有关的图文材料。

从软件的概念可以看出，程序并不是软件，程序只是软件的组成部分。

1.1.2　软件的特点

为了深入理解软件工程，探讨软件的特点是非常重要的。通过对软件特点的介绍，能使读者更好地理解计算机软件，能更充分地认识到软件工程的重要性。软件的特点可归纳如下：

（1）软件是一种逻辑实体。人们可以把它记录在介质上，但却无法看到软件的形态，而必须通过测试、分析、思考、判断去了解它的功能、性能及其他特性。软件正确与否，是好是坏，一直要到程序在机器上运行后才能知道，这就给软件的设计、生产和管理带来许多困难。

（2）软件开发是人类智力的高度发挥，而不是传统意义上的硬件制造。在软件的开发过程中没有明显的制造过程。软件是通过人们的智力活动，把知识与技术转化成信息的一种产品。所以，要实现对软件的质量控制，必须着重在软件开发方面下功夫。

（3）软件维护与硬件维修有着本质的差别。在软件的生存期中，为了使软件能够克服以前没有发现的故障，能够适应硬件、软件环境的变化以及用户新的要求，必须修改软件，而每次修改都可能会引入新的错误。这样反复修改软件，必然导致软件失效率升高。

（4）软件的开发和运行常常受到计算机系统的限制，对计算机系统有着不同程度的依赖性。为了解除这种依赖性，在软件开发中提出了软件移植的问题，并且把软件的可移植性作为衡量软件质量的因素之一。

（5）软件的开发至今尚未完全摆脱手工艺的开发方式，使软件的开发效率受到很大限制。因此，应加快软件技术的发展，提出和采用新的软件开发方法。例如，可利用软件复用技术或软件自动生成技术，使用一些有效的软件开发工具或软件开发环境，以提高软件开发效率。

（6）软件的开发是一个复杂的过程。有人认为，人类能够创造的最复杂的事物就是计

算机软件。软件的复杂可能来自它所反映的实际问题的复杂性，也可能来自程序逻辑结构的复杂性。

（7）软件的成本非常高昂。软件的研制工作需要投入大量的、复杂的、高强度的脑力劳动，所以开发费用非常高，导致软件的成本相当昂贵。

1.1.3 软件的分类

人们在学习、工作和生活中会接触到各种各样的软件。那么究竟有哪些类型的软件呢？虽然找不到一个统一的严格分类标准，但可以从不同角度来对软件进行分类。

1. 基于软件功能的划分

（1）系统软件（System software）：是服务于其他程序的程序集，一般由计算机生产厂家配置，如操作系统、数据库管理系统、设备驱动程序以及通信处理程序等。如果没有系统软件的支持，计算机将难以发挥其功能，甚至无法工作。

（2）应用软件（Application software）：是在系统软件的基础上，为解决特定领域应用而开发的软件。按其性质不同可以分为商业处理软件、科学计算软件、计算机辅助设计软件、人工智能软件等。

（3）支撑软件：是系统软件和应用软件之间的支持软件，一般用来辅助和支持开发人员开发和维护应用软件，以提高软件的开发质量和生产率。常用的支撑软件包括需求分析工具、设计工具、编码工具、测试工具、维护工具和管理工具等。

2. 基于软件工作方式的划分

（1）实时处理软件：指在事件或数据产生时，立即处理并及时反馈信号，控制那些需要监测和控制的过程的软件。主要包括数据采集、分析、输出三部分，其处理时间是应严格限定的，如果在任何时间超出了这一限制，都将造成事故。

（2）分时软件：允许多个联机用户同时使用计算机的软件。系统把处理机时间轮流分配给各联机用户，使各用户都感到只有自己在使用计算机的软件。

（3）交互式软件：能实现人机通信的软件。这类软件接收用户给出的信息，但在时间上没有严格的限定，这种工作方式给予用户很大的灵活性。

（4）批处理软件：把一组输入作业或一批数据以成批处理的方式一次运行，按顺序逐个处理的软件。

除此以外，软件还有其他的一些分类的方法。按软件规模进行划分可分为小型软件、中型软件、大型软件、甚大型软件和极大型软件；按软件服务对象的范围可划分为定制软件和产品软件等。

1.2 软件的发展和软件危机

1.2.1 计算机系统的发展历程

迄今为止，计算机系统经历了四个不同的发展阶段。下面简要地介绍各个发展阶段的特点，使读者了解软件危机的产生和软件工程学诞生的历史背景。

20 世纪 60 年代中期以前，是计算机系统发展的早期时代，此时的软件发展为程序设计阶段。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，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。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 20 世纪 70 年代中期，是计算机系统发展的第二代，此时的软件发展为程序系统阶段。在这 10 年中，计算机技术有了很大的进步，多道程序、多用户系统、实时系统、在线存储技术是这一时期的主要特征。在这一时期出现了“软件作坊”，但“软件作坊”基本上沿用了早期形成的个体化软件开发方法。随着软件需求数量的急剧膨胀，软件维护工作的急剧增长，“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。

计算机系统发展的第三代是从 20 世纪 70 年代中期到 20 世纪 80 年代中期，此时的软件发展为软件工程阶段。这个时期的主要特点是微处理器出现并得到了广泛的应用，分布式系统极大地增加了计算机系统的复杂性，局域网、广域网和宽带数字通信对软件开发者提出了更高的要求。但是，这个时期的软件仍然主要在工业界和学术界应用，个人应用还很少。

从 20 世纪 80 年代中期至今，计算机系统发展已进入第四代。这时人们已经不再看重单台计算机和程序，人们感受到的是硬件和软件的综合效果。由复杂操作系统控制的强大的桌面机、局域网和广域网，与先进的应用软件相配合，已经成为当前计算机发展的主流。计算机体系结构已迅速地从集中的主机环境转变成分布的客户机／服务器（或浏览器／服务器）环境，面向对象技术已经在许多领域迅速地取代了传统的软件开发方法。软件产业在世界经济中已经占有举足轻重的地位。

1.2.2 软件危机

20 世纪 60 年代末、70 年代初，西方工业发达国家经历了一场“软件危机”。这场软件危机表现在以下两点：一方面，软件十分复杂，价格昂贵，供需差日益增大；另一方面，软件开发时又常常受挫，质量差，指定的进度表和完成日期很少能按时实现，研制过程很难管理，即软件的研制往往失去控制。我们称软件开发和维护过程中所中遇到的这一系列严重问题为软件危机。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。

具体来说，软件危机主要有以下一些表现：

（1）软件功能不能满足用户的实际需要。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”必然导致用户对软件产品的不满意。

（2）软件产品的质量差。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。

（3）软件开发生产率低。软件生产率提高的速度远远不能满足客观需要，也跟不上硬件的发展速度，使人们不能充分利用现代计算机硬件所提供的巨大潜力。

（4）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。

（5）软件无配套的文档资料。软件开发人员可以利用文档资料，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。缺乏完整、合格的文档资料，必然给软件开发和维护带来许多严重的困难和问题。

（6）软件的可维护性差。很多程序中的错误非常难改正，并且不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。

（7）软件的价格昂贵。软件成本在计算机系统总成本中所占的比例逐年上升。

1.3 软件工程

1.3.1 软件工程的定义

要克服软件危机，软件开发必须寻找新的方法，创造新的理论。人们从失败中吸取教训，经过思索，得出如下结论：过去软件由同一个（或同一组）人编写和运行，如果出了故障也由同一个（或同一组）人来排除，这种手工作坊式的软件生产方式已经过时；软件不能再是个人艺术作品，软件开发人员不能是艺术家，软件开发人员应该是工程师。

有人从制造一台机器或修建一座楼房的过程中受到启发，提出：可否像机械工程、建筑工程那样，有计划、有步骤、守纪律地开展软件的开发工作？回答是肯定的，于是萌生了一种想法：像处理“工程”问题一样来处理软件开发全过程，这就产生了“软件工程”这一概念。“软件工程”一词于 1968 年问世以来，经过许许多多研究人员卓有成效的研究，终于产生了一门新兴的学科——软件工程学。

软件工程是指研究软件生产的一门学科，也就是将完善的工程原理应用于经济地生产既可靠又能在实际机器上有效运行的软件。1983 年美国《IEEE 软件工程标准术语》对软件工程下的定义为：软件工程是开发、运行、维护和修复软件的系统方法，其中“软件”的定义为计算机程序、方法、规则、相关的文档资料以及在计算机上运行时所必需的数据。

1.3.2 软件工程方法学

通常把在软件生命周期全过程中使用的一整套技术的集合称为软件工程方法学。软件工程方法学包括三个要素：方法、工具和过程。

软件工程方法是完成软件开发的各项任务的技术方法，为软件开发提供了“如何做”的技术。它包括了多方面的任务，如项目计划与估算、软件系统需求分析、数据结构、系统总体结构的设计、算法的设计、编码、测试以及维护等。软件工程方法中常采用某种特殊的语言或图形表达方法和一套质量保证标准。

软件工具为软件工程方法提供了自动的或半自动的软件支撑环境。目前已经开发出了许多软件工具，可以用于支持上述的软件工程方法。而且已经有人把诸多软件工具集成起来，使得一种工具产生的信息可以为其他的工具所使用，这样就建立起一种称之为计算机辅助软件工程（CASE）的软件开发支撑系统。

软件工程的过程则是将软件工程的方法和工具综合起来，以达到合理、及时地进行计算机软件开发的目的。过程定义了方法使用的顺序、要求交付的文档资料、为保证质量和协调变化所需要的管理以及软件开发各个阶段完成的里程碑。

传统方法学和面向对象方法学是目前使用得最广泛的两种软件工程方法学。本书后面

各章将对传统方法学进行详细的介绍，并简要介绍面向对象方法学。以下先对这两种方法学做一介绍和对比。

传统方法学也称为生命周期方法学或结构化范型，它采用结构化技术来完成软件开发的各项任务，并使用适当的软件工具或软件工程环境来支持结构化技术的运用。这种方法学把软件生命周期的全过程依次划分为若干个阶段，然后顺序地逐步完成每个阶段的任务。采用这种方法学开发软件的时候，从对任务的抽象逻辑分析开始，分阶段地进行开发。前一个阶段任务的完成是开始进行后一个阶段工作的前提和基础，而后一阶段任务的完成通常是使前一阶段提出的解法更进一步具体化，加进了更多的实现细节。每一个阶段的开始和结束都有严格标准，对于任何两个相邻的阶段而言，前一阶段的结束标准就是后一阶段的开始标准。在每一个阶段结束之前都必须进行正式严格的技术审查和管理复审，从技术和管理两方面对这个阶段的开发成果进行检查，通过之后这个阶段才算结束；如果检查通不过，就必须进行必要的返工，并且返工后还要再经过审查。审查的一条主要标准就是每个阶段都应该交出“最新式的”（即和所开发的软件完全一致的）高质量的文档资料，从而保护在软件开发工程结束时有一个完整准确的软件配置交付使用。在完成生命周期每个阶段的任务时，应该采用适合该阶段任务特点的系统化的技术方法——结构化分析、结构化设计、结构化程序设计或结构化测试技术。

传统方法学的优点在于：把软件生命周期划分成若干个阶段，每个阶段的任务相对独立，而且比较简单，便于不同人员分工协作，从而降低了整个软件开发工程的困难程度；在软件生命周期的每个阶段都采用科学的管理技术和良好的技术方法，而且在每个阶段结束之前都从技术和管理两个角度进行严格的审查，合格之后才开始下一阶段的工作，这就使软件开发工程的全过程以一种有条不紊的方式进行，保证了软件的质量，特别是提高了软件的可维护性。总之，采用传统方法学可以大大提高软件开发的成功率和软件开发的生产率。

但是传统方法学并不适合于以下情况：软件规模庞大，或者对软件的需求是模糊的或随时间变化的。同时，使用传统方法学开发出来的软件，可维护性并不是最佳的。归结原因，在于传统方法学存在以下局限：这种技术要么面向行为（即对数据的操作），要么面向数据，没有既面向数据又面向行为的结构化技术。同时，离开了操作便无法更改数据，而脱离了数据的操作更是毫无意义的，数据和对数据的处理原本就是密切相关的。然而，传统方法学把数据和对数据的处理人为地分离成两个独立的部分，自然增加了软件开发与维护的难度。

为了克服传统方法学的局限，逐步发展并形成了一种新的方法学——面向对象方法学。面向对象方法把数据和行为看成同等重要，它是一种以数据为主线，把数据和对数据的操作紧密地结合在一起的方法学。

面向对象方法具有以下四个要点：

（1）把对象作为融合了数据以及在数据上的操作行为的统一的软件构件。面向对象的程序是由对象组成的，程序中任何元素都是对象，复杂对象由比较简单的对象组合而成。

（2）把所有对象都划分成类。每个类都定义了一组数据和一组操作，类是对具有相同数据和相同操作的一组相似对象的定义。数据用于表示对象的静态属性，是对象的状态信息，而施加于数据之上的操作用于实现对象的动态行为。

（3）根据基类与派生类的关系，把若干个相关类组成一个层次结构的系统。在类的等级中，下层派生类自动拥有上层基类所定义的数据和操作，这种现象称为继承。

（4）对象之间只能通过发送消息相互联系。对象与传统数据有着本质的区别：对象不

是被动地等待外界对它施加操作，相反，对象是进行处理的主体，必须向它发消息请求它执行它的某个操作以处理它的数据，而不能从外界直接对它的数据进行处理。也就是说，对象的所有私有信息都被封装在该对象内，不能从外界直接访问，这就是通常所说的封装性。

面向对象方法学的出发点和基本原则，是尽可能模拟人类习惯的思维方式、方法与过程，尽可能接近人类认识世界、解决问题的方法与过程，从而使描述问题的问题空间（也称为问题域）与实现解法的解空间（也称为求解域）在结构上尽可能一致。用面向对象方法学开发软件的过程，是一个主动地多次反复迭代的演化过程。面向对象方法在概念和表示方法上的一致性，也保证了在各项开发活动之间的平滑（无缝）过渡。

正确地运用面向对象方法学开发软件，最终的软件产品由许多较小的基本上独立的对象组成，而且大多数对象都与现实世界中的实体相对应，因此，降低了软件产品的复杂性，提高了软件产品的可理解性，简化了软件的开发和维护工作。由于对象是相对独立的实体，容易在以后的软件产品中重复使用，因此，与传统方法学相比，面向对象方法学促进了软件的重用。同时，面向对象方法学中对象所特有的继承性，又使软件的可重用性进一步提高。

1.4 软件生存期和软件开发模型

1.4.1 软件生存期

如同任何其他事物一样，软件也有一个孕育、诞生、成长、成熟、衰亡的生存过程，一般称之为计算机软件的生存期。

一般说来，软件生命期由软件定义、软件开发和软件维护三个时期组成，每个时期又可进一步划分成若干个阶段。

1. 软件定义时期

（1）问题定义：这是软件生存期的第一个阶段，主要任务是弄清用户要计算机解决的问题是什么。通过问题定义阶段的工作，系统分析员应该提出关于问题性质、工程目标和规模的书面报告。

（2）可行性研究：任务是为前一阶段提出的问题寻求一种至数种在技术上可行且在经济上有较高效益的解决方案。可行性研究阶段应该导出系统的高层逻辑模型（通常用数据流图表示），并且在此基础上更准确、更具体地确定工程的规模和目标。然后由系统分析员更准确地估计系统的成本和效益，对系统进行仔细的成本／效益分析。同时，还应制订出人力、资源及进度计划。

2. 软件开发时期

（1）需求分析：任务在于弄清用户对软件系统的全部需求，主要是确定目标系统必须具备哪些功能。系统分析员在需求分析阶段必须和用户密切配合，充分交流信息，以得出经过用户确认的系统逻辑模型。通常用数据流图、数据字典和简要的算法表示系统的逻辑模型。这些文档既是软件系统逻辑模型的描述，也是下一阶段进行设计的依据。

（2）总体设计：任务是设计软件的结构，即确定程序由哪些模块组成以及模块间的关系。通常用层次图或结构图描绘软件的结构。

（3）详细设计：任务是针对单个模块的设计。目的是确定模块内部的过程结构，并设计出程序的详细规格说明。通常用 HIPO 图（层次图加输入／处理／输出图)或 PDL 语言（过程设计语言）描述详细设计的结果。

（4）编码：任务是按照选定的语言，把模块的过程性描述翻译为源程序。这个阶段的关键是，程序员根据目标系统的性质和实际环境，选取一种适当的程序设计语言，并把详细设计的结果翻译成用选定的语言书写的正确的、容易理解、容易维护的程序模块。

请读者注意，直到这一阶段，才产生了能在计算机上执行的源程序。

（5）测试：测试是开发时期的最后一个阶段，这个阶段的任务是通过各种类型的测试（及相应的调试）使软件达到预定的要求。应该用正式的文档资料把测试计划、详细测试方案以及实际测试结果保存下来，作为软件配置的一个组成部分。

3. 软件运行时期

软件运行时期是软件生存周期的最后一个时期。软件人员在这一时期的工作主要是做好软件维护。维护的目的是使软件在整个生存周期内保证满足用户的需求和延长软件的使用寿命。实际上，每一项维护实质上是经历了一次压缩和简化了的软件定义、开发的全过程。每一项维护活动都应该准确地记录下来，作为正式的文档资料加以保存。

以上简要地介绍了软件生存周期各个阶段的主要任务和评审标准。本书在以后各章中将围绕软件生存周期的各个阶段，详细讲述每个阶段所要完成的任务、所需的技术方法、辅助工具以及软件开发和维护的各种管理技术。

1.4.2 软件开发模型

软件开发模型是软件开发中全部过程、活动和任务的结构框架，是软件开发工作的基础。软件开发模型能清晰、直观地表达软件开发全部过程，明确地规定要完成的主要活动和任务。软件开发模型都应该是稳定和普遍适用的。

软件开发包括需求、设计、编码和测试等阶段，有时也包括维护阶段。软件开发模型对于不同的应用系统，允许采用不同的开发手段和方法，使用各种不同的程序设计语言以及各种不同技能的人员参与工作，还应允许采用不同的软件工具或各种不同的软件工程环境。

最早出现的软件开发模型是 1970 年 W.Royce 提出的瀑布模型，直到现在它仍然是软件工程中使用最广泛的过程模型。而后随着软件工程学科的发展和软件开发的实践，相继提出了螺旋模型、喷泉模型等。

1. 瀑布模型

瀑布模型将软件生存周期的各项活动规定为依照固定顺序连接的若干阶段工作，形如瀑布流水，最终得到软件产品。初始瀑布模型如图 1.1 所示。

这一模型规定了开发各阶段的活动为，提出系统需求、提出软件需求、需求分析、设计、编码、测试和运行，并且还规定了自上而下相互衔接的固定顺序，于是构成了人们熟知的瀑布模型。然而实践表明，各开发阶段间的关系并非完全是自上而下的线性因式，每个开发阶段均具有以下特征：

（1）从上一阶段接受本阶段工作的对象，作为输入；

（2）对上述输入实施本阶段的活动；

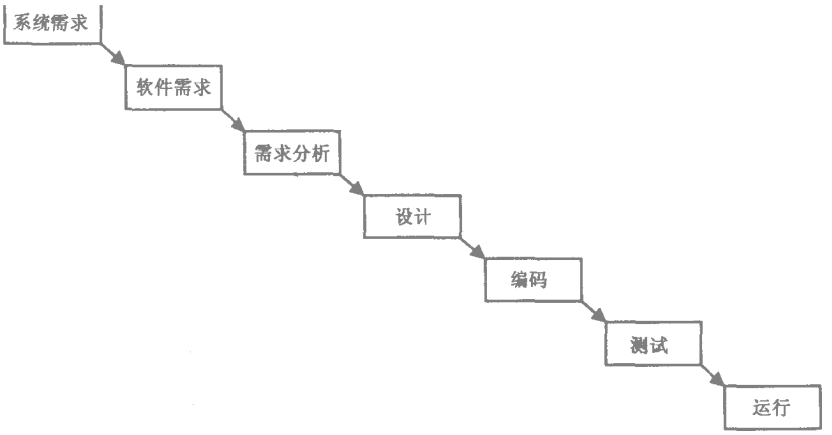

图 1.1 初始瀑布模型

（3） 给出本阶段的工作成果，作为输出传入下一阶段；

（4） 对本阶段工作进行评审，若本阶段的工作得到确认，则继续下阶段工作，否则返回前一阶段，甚至更前阶段。

为表达向前阶段的反馈，在模型图中增加了虚线表示的箭头，从而构成了具有反馈回路的瀑布模型,如图 1.2 所示。

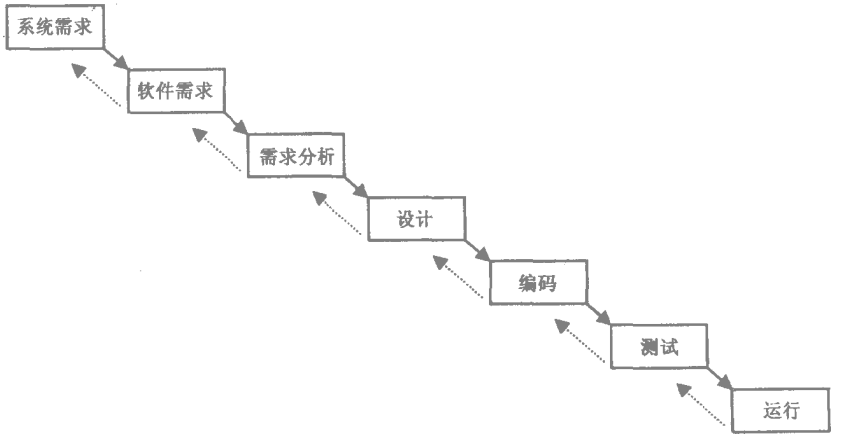

图 1.2 具有反馈回路的瀑布模型

许多采用瀑布模型的开发组织为更有效地组织实施，制定了软件开发规范或开发标准。在这些软件开发规范或开发标准中，明确规定了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。瀑布模型自诞生以来之所以广泛流行，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的作用显著。

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就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。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，无疑要进行返工，甚至不得不在维护阶段中纠正需求的偏差，为此必然要付出高昂的代价，会造成许多不必要的损失。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突出。

2. 螺旋模型

为了克服瀑布模型的不足，螺旋模型于 1988 年由 B.Boehem 提出。该模型中加入了风险分析，通常用来指导大型软件项目的开发。

软件风险是任何软件开发项目中普遍存在的问题，不同项目其风险有大有小。在制定软件开发计划时，系统分析员必须回答“项目的需求是什么”、“需要投入多少资源”以及“如何安排开发进度”等一系列问题。然而若要当即给出准确无误的回答是不容易的，甚至几乎是不可能的。但系统分析员又不可能完全回避这一问题。单单凭借经验进行估计难免带来一定风险。实践表明，项目规模越大、问题越复杂，资源、成本、进度等因素的不确定性就越大，承担项目所冒的风险也越大。风险是软件开发中不可忽视的潜在不利因素，它可能在不同程度上损害到软件开发过程和软件产品的质量。软件风险驾驭的目标是在造成危害之前，及时对风险进行识别和分析，以采取适当对策，进而消除或减少风险的损害。螺旋模型如图 1.3 所示。

沿着螺线旋转，在笛卡尔坐标的四个象限上分别表达了四个方面的活动，即

（1）制定计划——确定软件目标，选定实施方案，弄清项目开发的限制条件

（2）风险分析——分析所选方案，考虑如何识别和消除风险；

（3）实施工程——实施软件开发；

（4）客户评估— 评价开发工作，提出修正建议。

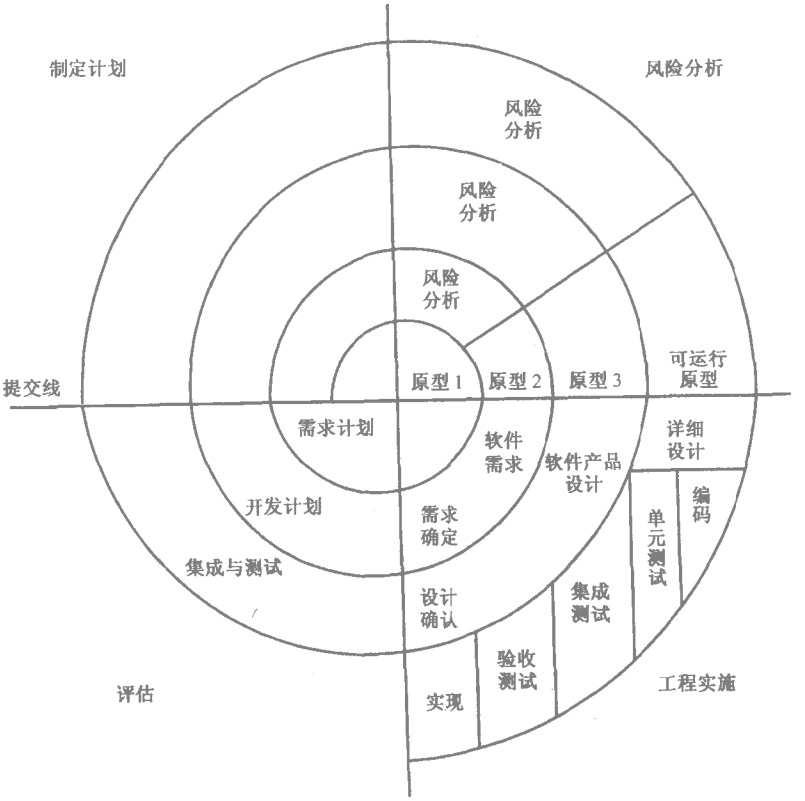

图 1.3 螺旋模型

沿螺线自内向外每旋转一圈便开发出更为完善的一个新的软件版本。例如，在第一圈确定了初步的目标、方案和限制条件以后，转入右上象限，对风险进行识别和分析。如果风险分析表明，需求具有不确定性，那么在右下的工程象限内，所建的原型会帮助开发人员和客户考虑其他开发模型，并将需求做进一步修正。客户对工程成果做出评价后，给出修正建议，在此基础上需再次计划，并进行风险分析。在每一圈螺线上，风险分析的终点做出是否继续下去的判断。假如风险过大，开发者和用户无法承受，项目有可能终止。多数情况下沿螺线的活动会继续下去，自内向外逐步延伸，最终得到所期望的系统。

3. 喷泉模型

喷泉模型体现了软件创建所固有的迭代和无间隙的特征。喷泉模型如图 1.4 所示。

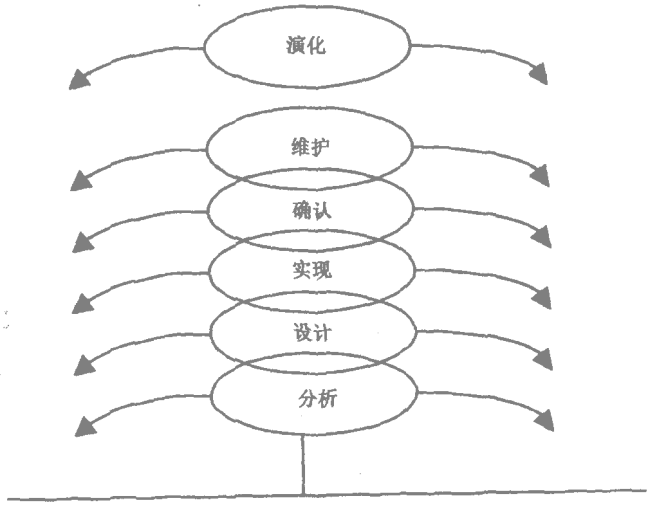

图 1.4　喷泉模型

这一模型表明了软件刻画活动需要多次重复。例如，在编码之前，再次进行分析和设计，其间，添加有关功能并使系统得以演化。同时，该模型还表明活动之间没有明显的间隙，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。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，并且这一表达自然地支持复用。

小　　结

本章首先介绍了软件的概念，并讨论了软件的特点和分类。通过对计算机系统的发展历程的介绍，使读者了解软件危机产生和软件工程学诞生的历史背景，并列举了软件危机的主要表现。然后引出了软件工程的概念，并对目前使用广泛的软件工程方法学，即传统方法学和面向对象方法学进行了简要介绍和对比。最后介绍了软件生存期的八个阶段，并讨论了瀑布模型、螺旋模型、喷泉模型等软件开发模型。

习 题 1

1.1 什么是软件？软件和程序有何区别？
1.2 简述软件的特点和分类。
1.3 什么是软件危机？软件危机有哪些表现？
1.4 什么是软件工程？
1.5 什么是软件生存期？软件生存期分哪几个阶段？
1.6 试比较瀑布模型、螺旋模型和喷泉模型。

第 2 章　可行性研究

2.1　问题定义

当用户提出软件研制的要求时，软件生命周期就开始了。问题定义阶段必须回答的关键问题是“要解决的问题是什么”。如果在问题尚未完全明确的情况下就试图解决这个问题，那么只会白白地浪费时间和财物，最终得到的结果很可能是毫无意义的。尽管确切地定义问题的必要性十分明显，但在实际中它却是最容易被忽视的一个步骤。

2.1.1　问题定义的内容

问题定义的内容包括明确问题的背景、开发系统的现状、开发的理由和条件、开发系统的问题要求、总体要求、问题的性质、类型范围、要实现的目标、功能规模、实现目标的方案、开发的条件、环境要求等，然后写出问题定义报告（或称系统定义报告），以供可行性分析阶段使用。

2.1.2　问题定义的步骤

在问题定义阶段，系统分析员要深入现场，阅读用户写的书面报告，听取用户对开发系统的要求，调查开发系统的背景理由。还要与用户负责人反复讨论，以澄清模糊的地方，改正不正确的地方。最后写出双方都满意的问题定义报告，并确定双方是否可进行深入系统可行性研究的意向。

下面是问题定义报告的例子。

某校教材科提出开发微机教材销售系统的要求，经过系统分析员的调查，写出如下的问题定义报告，说明微机教材销售系统的目标范围。

（1）项目：教材销售系统。

（2）背景：人工销售效率低，易出错。

（3）项目目标：建立一个高效率的、无差错的微机教材销售系统。

（4）项目范围：硬件利用现有微机，软件开发费不超过 5 000 元。

（5）初步设想：增加缺书统计与采购功能。

（6）可行性研究：建议进行一周，费用不超过 500 元

____年____月____日　签字：_____

2.2　可行性研究的任务

可行性研究的任务是用最小的代价、在尽可能短的时间内确定问题是否能够解决。必须注意的是，可行性研究的目的并不是解决问题，而是确定问题是否值得去解决，也就是判断系统原定的目标和规模是否现实，通过实际使用完成后的系统所能带来的效益是否大到值

得投资开发这个系统的程度。因此，可行性研究实质上是要进行一次大大压缩和简化了的系统分析和设计的过程，也就是在较高层次上以较抽象的方式进行的系统分析和设计的过程。

在澄清了问题定义之后，系统分析员首先应该导出系统的逻辑模型，然后从系统逻辑模型出发，探索出若干种可供选择的主要解法（即系统实现方案），最后仔细研究每种解法的可行性。

一般说来，研究可行性应该从下述几方面进行。

（1）技术可行性：指使用现有的技术能否完成这个项目。例如，现有计算机能否达到所需的信息处理速度，现有的输入／输出设备能否承担所要求的数据输入／输出量，开发项目所需的硬、软件资源能否按期到达，有无胜任开发系统的熟练技术人员等。

（2）经济可行性：指通过对软件开发项目进行成本／效益估计，以确定软件系统可能带来的经济效益能否超过研制和维护此系统所需的费用。

（3）社会因素的考虑：软件开发是否会侵犯他人、集体或国家的利益，是否违反国家的法律并可能由此而承担法律责任。例如，某项目如果采用微波技术作为通信手段来传送信息，那么价格便宜并且技术可行。但是由于保密方面的要求，国家明文规定在大城市禁止使用微波。因此，方案中的微波通信就不能采用，可改用电缆来进行通信。

当然，可行性研究最根本的任务是对以后的行动方针提出建议。如果通过进行可行性研究，表明问题没有可行的解，那么系统分析员应该建议停止这项开发工程，以避免造成时间、资源、人力和金钱的浪费；如果结论是问题值得去解决，那么系统分析员应该推荐一个较好的系统实现方案，并且为此项工程制定一个初步的开发计划。

2.3 可行性研究的步骤

一般地说，可行性研究的步骤如下。

（1）复查系统规模和目标。系统分析员仔细地阅读和分析有关的材料，以便进一步复查确认系统规模和目标，改正含糊或不确切的叙述，清晰地描述对目标系统的一切限制和约束，即确保系统分析员正在解决的问题确实是要求他解决的问题。

（2）研究目前正在使用的系统。现有的系统是信息的重要来源，运行旧系统所需要的费用是一个重要的经济指标。如果新系统不能增加收入或减少使用费用，那么从经济角度来看新系统就不如旧系统。

邀请有关人员检验系统分析员对现有系统的认识是否正确，但注意不要花费太多时间去了解现有系统的实现细节。例如，除非是为了阐明一个特别关键的算法，否则不需要根据代码画出程序流程图。没有一个系统是在“真空”中运行的，绝大多数系统都与其他系统有联系。所以还应该注意了解并记录现有系统和其他系统之间的接口情况，这是设计新系统时的重要约束条件。

（3）导出新系统的高层逻辑模型。优秀的设计通常总是从现有的物理系统出发，通过参考现有系统的逻辑模型来设想目标系统的逻辑模型，最后根据目标系统的逻辑模型建造新的物理系统。

系统分析员对目标系统有了一定的了解后，可以使用数据流图描绘数据在系统中流动和处理的情况，从而概括地表达出对新系统的设想。为了把新系统描绘得更清晰准确，还应该有一个初步的数据字典，定义系统中使用的数据。数据流图和数据字典共同定义了新系统

的逻辑模型，以后可以从这个逻辑模型出发进行新系统的设计。

（4）重新定义问题。新系统的逻辑模型实质上表达了系统分析员对新系统必须做什么的看法。那么用户是否也有同样的看法呢？系统分析员应该和用户一起对问题定义、工程规模和目标进行再次复查。这次复查应该把数据流图和数据字典作为讨论的基础。如果系统分析员对问题有误解或者用户遗漏了某些要求，那么应及时发现并进行改正。

可行性研究的前四个步骤实质上构成了一个循环。系统分析员定义问题、分析这个问题、导出一个试探性的解，在此基础上再次定义问题、再次分析、再次修改……继续这个过程，直到提出的逻辑模型完全符合系统目标为止。

（5）导出和评价供选择的方案。系统分析员从系统逻辑模型出发，导出若干个较高层次的（较抽象的）物理解法供比较和选择。当从技术角度提出了一些可能的物理系统之后，首先，应该根据技术可行性的考虑，初步排除一些不现实的系统。其次，可以考虑经济方面的可行性，系统分析员应该估计余下的每个可能的系统的开发成本和运行费用，并且估计相对于现有的系统而言这个系统可以节省的开支或可以增加的收入。在这些估计数字的基础上，对每个可能的系统进行成本／效益分析。最后，可以考虑社会（法律）可行性。系统分析员分析开发系统是否符合当前社会生产管理经营体制要求，有无版权纠纷、生产安全以及与国家法律相违背的问题，以做出社会（法律）可行性的结论。

（6）推荐方案和行动方针。根据可行性研究结果，如果系统分析员认为值得继续进行这项工程的开发，那么他应该选择出一种最好的解决方案，并且说明选择这个解决方案的理由。

由于使用部门负责人通常主要是根据经济上是否划算来决定是否投资于某一项开发项目，因此系统分析员对于所推荐的方案必须仔细地进行成本／效益分析，以此作为决定是否投资的依据。

（7）草拟开发计划。系统分析员进一步为推荐的系统草拟一份开发计划。除了工程进度表之外，还应该估计对各种开发人员（包括系统分析员、程序员、资料员等）和各种资源（计算机硬件、软件工具等）的需要情况，并且指明这些人员和资源什么时候使用、使用多长时间。此外，对系统生命周期中每个阶段的成本进行估计。最后，给出下一个阶段（即需求分析阶段）的详细进度表和成本估计。

（8）书写文档，提交审查。把上述可行性研究各个步骤得到的结果写成清晰的文档，并请用户和使用部门的负责人仔细审查，以决定是否继续这项工程，是否接受系统分析员所推荐的解决方案。

2.4 系统流程图

在进行可行性研究时需要了解和分析现有的系统，并以概括的形式表达对现有系统的认识；进入设计阶段以后应该把设想的新系统的逻辑模型转变成物理模型，因此需要描绘未来的物理系统的概貌。

可行性研究对现有系统做概括的物理模型描述，如用图形工具表示则非常直观和简洁。系统流程图是描绘物理系统的传统工具，它的基本思想是用图形符号以黑盒子形式描绘系统里面的每一个部件（程序、文件、数据库、表格、人工过程等）。需要注意的是，尽管系统流程图使用的某些符号和程序流程图所用的符号相同，但系统流程图表达的是信息在系统中各个部件之间流动的情况，而不是对信息进行加工处理的控制过程。系统流程图的基本符号

见表 2.1 .

表 2.1 系统流程图的基本符号

符　号	名　称	说　明
	处理	能改变数据值或数据位置的加工或部件
	输入/输出	表示输入或输出（或既输入又输出），是一个广义的不指明具体设备的符号
	连接	指出转到图的另一部分或从图的另一部分转来，通常在同一页上
	换页连接	指出转到另一页图上或由另一页图转来
	人工操作	由人工完成处理
	数据流	用来连接其他符号，指明数据流动方向

例 2.1　用系统流程图分析下述问题。

某图书超市仓库的管理如下：仓库中现有各种图书的数量和每种图书的库存量临界值等数据记录在库存清单主文件中。当仓库中图书数量等数据有所变化时，应该及时修改库存清单主文件。当某种图书的库存量少于它的库存量临界值时，则应该将信息报告给采购部门以便订货。

如图 2.1 所示的系统流程图描述了上述系统的概貌。图中的每个符号定义了组成系统的一个部件，而并没有指明每个部件的具体工作过程。图中的箭头指定了系统中信息流动的路径。

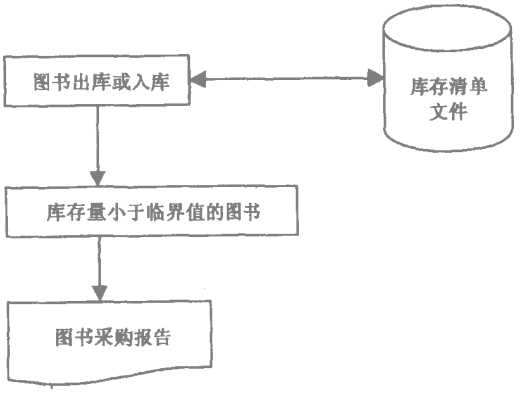

图 2.1　图书超市仓库系统流程图

2.5　成本／效益分析

成本／效益分析的目的是从经济角度评价开发一个新项目是否可行、是否划算，从而帮助使用部门的负责人正确地做出是否投资于这项开发的决定。一般说来，人们投资于一项事业的目的是为了在将来得到更大好处。开发一个系统也是一种投资，同样期望将来获得更大的经济效益。经济效益通常表现为减少运行费用或增加收入。进行成本／效益分析时，首先要估算待开发系统的开发成本，然后与可能取得的效益（有形的和无形的效益）进行比较和

权衡。其中有形的效益可用货币的时间价值、投资的回收期、纯收入等指标进行度量；无形的效益主要是从性质上、心理上进行衡量，很难直接进行量的比较。但是，无形的效益有特殊的潜在价值，且在某些情况下可能转化为有形的效益。

2.5.1 成本估计

由于人、技术、环境、政治等众多因素，都可能影响到软件开发的成本，所以成本估计是一门不很精确的技术。在实际应用中，应该同时使用几种不同的成本估计技术，以便相互校验。下面简单介绍两种成本估计技术。

1. 代码行技术

代码行技术是一种比较简单的定量估算成本的方法，它把开发每个软件功能的成本和实现这个功能所需要使用的源代码行数联系起来。通常根据经验和历史数据来估计实现一个功能所需要的源程序行数。在有以往开发类似工程的历史数据供参考的情况下，这个方法非常有效。

一旦估计出源代码行数以后，用每行代码的平均成本乘以行数就可以确定软件的成本。每行代码的平均成本主要取决于软件的复杂程度和开发人员的工资水平。

2. 任务分解技术

任务分解技术首先把软件开发工程分解为若干个相对独立的任务，再分别估计每个单独的开发任务的成本，最后累加起来得出软件开发工程的总成本。估计每个任务的成本时，通常先估计完成该项任务所需要使用的人力（以人月为单位），再乘以每人每月的平均工资而得出每个任务的成本。划分任务时最常用的办法是按开发阶段进行。如果软件系统很复杂，由若干个子系统组成，则可以把每个子系统再按开发阶段进一步划分成更小的任务。

典型环境下各个开发阶段需要使用的人力的百分比如表 2.2 所示。应该针对每个开发工程的具体特点并参照以往的经验，尽可能准确地估计每个阶段实际所需要使用的人力（包括书写文档需要的人力）。

表 2.2 典型环境下各个开发阶段所需人力的百分比

任　务	所占人力（%）
可行性研究	5
需求分析	10
软件设计	25
编码和单元测试	20
综合测试	40
总计	100

2.5.2 度量效益的方法

成本／效益分析应包括估计开发成本、运行费用和新系统将带来的经济效益。运行费用取决于系统的操作费用（操作员人数、工作时间、消耗的物资等）和维护费用。系统的经济效益则等于因使用新系统而增加的收入加上使用新系统可以节省的运行费用。

下面介绍几种度量效益的方法。

1. 货币的时间价值

在比较新系统的开发成本和经济效益时，应该考虑到的问题是，投资是现在进行的，而效益是将来获得的，所以不能只是简单地对成本和效益进行比较，而是应该考虑货币的时间价值。

货币的时间价值通常用利率的形式表示。假设年利率为 i，如果现在存入 P 元，则 n 年后可以得到的钱数为：

$$F=P\times(1+i)^n$$

这也就是 P 元钱在 n 年后的价值。反之，如果 n 年后能收入 F 元钱，那么这些钱的现在的价值是：

$$P=F\div(1+i)^n$$

例如，修改一个已有的图书超市库存清单系统，使它能在每天送给采购员一份订货报表。修改已有的库存清单程序并且编写产生报表的程序，估计共需 1.5 万元；系统修改后能及时订货以消除图书短缺问题，估计因此每年可以节省 0.6 万元，五年共可节省 3 万元。但是，不能简单地把 1.5 万元和 3 万元相比较，因为前者是现在投资的钱，后者是若干年以后节省的钱。

假定年利率为 5%，利用上面计算货币现在价值的公式，可以算出修改库存清单系统后每年预计节省的钱的现在价值，参见表 2.3。

表 2.3　货币的时间价值

年　份	将来值（万元）	$(1+i)^n$	现在值（万元）	累计的现在值（万元）
1	0.6	1.05	0.5714	0.5714
2	0.6	1.1025	0.5442	1.1156
3	0.6	1.1576	0.5183	1.6339
4	0.6	1.2155	0.4936	2.1275
5	0.6	1.2763	0.4701	2.5976

2. 投资回收期

通常用投资回收期来衡量一项开发工程的价值。所谓投资回收期就是使累计的经济效益等于最初投资所需要的时间。显然，投资回收期越短就能越快获得利润，这项工程也就越值得投资。

上例中，修改库存清单系统两年以后可以节省 1.1156 万元，比最初的投资（1.5 元）还少 0.3844 万元。第三年中可再节省 0.5183 万元，则：

$$0.3844\div0.5183=0.748$$

因此，投资回收期是 2.748 年。

3. 纯收入

衡量工程价值的另一项经济指标是工程的纯收入，也就是在整个生命周期之内系统累计经济效益（折合成现在值）与投资之差。这相当于比较投资开发一个软件系统和把钱存在银行中（或贷给其他企业）这两种方案的优劣。如果纯收入为零，则工程的预期效益和在银行存款一样，但是开发一个系统要冒风险，因此从经济观点看这项工程可能是不值得投资的。