第 1 章　绪 论

在人类认识和征服自然的过程中，对于工作量大、涉及人员多的工作，往往需事先制订计划，然后按计划进行，这便是工程。秦昭襄王 51 年（公元前 256 年），李冰在其子二郎的协助下，邀集有治水经验的当地人民，对岷水东流的地形和水情作了实地勘察，通过实施引水、分水和排砂，消除了水患，使川西平原变成“天府之国”。二千二百多年来，都江堰水利工程至今仍然连续使用，仍发挥着巨大的效益。就软件而言，随着计算机应用的深入，其开发和维护的工作量愈来愈大、涉及的人员愈来愈多。同水利工程有计划地实施水利开发活动一样，软件工程也通过实施一系列有计划的软件开发和维护活动来进行，其目的在于以较少的代价来获取高质量的软件。

本章主要使读者了解软件工程的概貌，其内容包括：软件工程学科的形成，软件工程的研究对象,软件生存期过程与模型,软件工程标准、方法、工具和环境。

1.1　软件及其发展

计算机系统由软件和硬件这两个密不可分的部分组成。硬件是系统中看得见、摸得着的物理设备。硬件的发展从电子管到晶体管，再到集成电路，直至超大规模、超高速集成电路，性能越来越强，价格越来越便宜。软件是系统中看不见、摸不着的逻辑部分，以程序、数据和文档的形式出现。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操纵信息的数据结构；文档描述了与程序开发、使用和维护有关的文字或图形资料。硬件性能的提高，极大地促进了更为复杂的软件系统的出现。软件系统的复杂性给软件系统的开发和维护带来一系列问题，由此也推动软件不断向前发展。

1.1.1　软件特征

了解软件的特征并据此明白软件与硬件之间的区别对更好地理解软件的发

展是必要的。同硬件相同，软件是一种无形的逻辑实体，由此具有与硬件不同的特殊性质。

1.软件不是传统意义上的‘制造’产生的,而是‘研发’出来的

软件制造只是简单的拷贝，一旦软件项目研发成功，通过复制就可以产生大量的软件产品，因而软件的成本和质量主要取决于软件的研发。硬件的成本和质量则主要取决于硬件研制成功后的重复制造过程。虽然‘制造’和‘研发’都依赖于人,但参与者和完成的工作之间的关系不同。同时,尽管‘制造’和‘研发’的目标是构造‘产品”,但两者采用的方法不同。这意味着,软件项目的管理、软件产品的保护等等必须不同于硬件。

2.软件不会被“用坏”

软件和硬件一样，在使用的早期，会由于设计或研发、制造的错误或缺陷造成故障，但随时间的推移，由于错误和缺陷的改正，故障率逐步降低。在使用的后期，硬件会因为磨损和老化等问题，故障率逐步上升。当故障率达到一定程度后,硬件就会因‘用坏’而寿终,软件不会在使用过程中出现磨损和老化,因此软件不会因“用坏”而寿终。但软件会因环境或需求的变化必须多次修改（维护）。在实际中，软件的每次维护都不可避免地引入新的错误。同时，软件的维护过程实质是一次研发过程，不能仅通过软件的重复制造来解决。而硬件因磨损或老化等导致的“用坏”则可以通过重复制造的备用零件替换来解决。所以，软件的维护要远比硬件的维护复杂得多。

3.软件大多是‘定制’的

现代工业社会中，硬件基本都通过组装不同生产厂商的各种零部件而设计和制造出新产品，定制的成分比较少。软件至今很少能做到利用现成的部件通过组装形成所需的软件。软件大多是为用户专门“定制”而成的。对于软件开发人员而言，软件开发工作依然是一种高强度的脑力劳动。软件复用技术、自动生成技术等，在现有的软件项中使用的比率仍然很低，软件产品开发还未完全摆脱传统的手工开发方式。正因如此，软件开发的质量和效率受到很大的限制。

4.软件成本难于估计

硬件成本主要取决于研发后的制造过程，容易进行成本估算并制订周密计划。软件成本主要取决于软件的研发过程，一方面，软件需求和程序逻辑结构复杂，且其复杂性与软件技术的发展不相适应，同时软件技术发展落后的差距越来越大；另一方面，软件产品的研发需要大量的脑力劳动。因此，软件成本难于估计且越来越昂贵。人们对软件成本的不准确估计往往造成软件项目计划失效，进而直接影响软件项目的成败

1.1.2 软件分类

软件的分类从另一方面反映了软件的发展。前面从软件与硬件的比较中得出了各种软件的共同特点。但不同类型的软件，其开发和维护又有各自的不同特点。由于软件本身的复杂性，难以找到一个统一的严格分类标准。本节从各种不同的角度来对软件进行分类。

1.按应用功能分类

• 系统软件：与计算机系统硬件紧密交互，协调计算机系统各部分工作的软件。例如操作系统、设备驱动程序及通信处理程序等。系统软件是计算机系统必不可少的一个组成部分。

• 支持软件：协助使用者开发软件的工具性软件。例如程序编译器、自动化测试软件、系统分析辅助工具及软件开发管理工具等。

• 应用软件：为使一个计算机系统得到某种功能而专门开发的软件。例如：商业信息处理软件、工程和科学计算软件、人工智能软件及智能产品嵌入软件（如微波炉的按钮控制、汽车中的燃料控制等）、多媒体播放软件等。有时，支持软件和应用软件的划分边界比较模糊，如字处理软件，既是支持软件辅助软件开发，又可看成应用软件。

GB/T 13702－92 按计算机软件的基本特征和主要功能，采用线性分类法对软件进行了分类。对于每一种类型，给出了对应的代码，其表示形式如图 1－1所示。

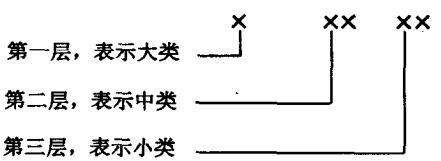

图 1－1

例如，系统软件、支持软件和应用软件是三大类，其编号分别为 1、3、6。操作系统软件的编号为 110，表示其属于系统软件大类；软件开发工具的编号为 310，表示其属于支持软件大类；测试工具的编号为 31040 表示其属于软件开发工具中类。

2.按服务对象的范围分类

• 项目软件：软件开发机构受特定用户委托开发而成的软件。例如，电信管理系统、空中交通管制系统、军用防空指挥系统、生产过程控制系统等。一般情况下，项目软件在合同的约束下开发。为了争取软件开发合同，软件开发机构必须重视质量管理，而软件开发机构的技术实力、开发经验以及社会信誉等也相当重要。

• 产品软件：软件开发机构直接为市场开发的软件。例如，字处理软件、多媒体播放软件、游戏软件、教育软件等等。产品软件的功能、性能、价格和售后服务对开发机构参与市场竞争有重要影响。

1.1.3 软件发展与软件危机

软件发展大致可以划分为三个阶段：程序设计阶段、程序系统阶段和软件工程阶段。

1.程序设计阶段

20世纪50至60年代，人们使用机器语言或汇编语言书写程序。由于硬件价格昂贵、存储容量小、运行可靠性差等，人们在编写程序时追求编程技巧、追求程序运行效率，程序设计被看作是一种发挥个人创造才能的技术领域。而且，当时人们认为写出的程序只要能在计算机上运行并得出正确的结果就行，程序的写法可以不受任何限制。这些氛围，使得编写的程序难于理解、修改困难。在程序设计阶段，软件的生产主要是专家的个体手工劳动。

2.程序系统阶段

在20世纪60至70年代，硬件在性能和价格方面有了明显的改善，计算机的应用领域不断扩大，软件的需求也由此不断增长。软件由于处理领域的扩大而变得复杂，软件的生产难于由单个个体完成，于是形成了“软件作坊”，使用高级程序设计语言来开发程序。在程序系统阶段，开发人员开始注重程序设计风格，提出了结构化程序设计方法与模块化设计思想。但“软件作坊”基本上还是沿用程序设计阶段形成的个体手工开发方式。在此阶段，不仅大量的软件需要开发和维护；同时，已有的软件开发和维护技术以及组织管理不适应大规模复杂软件的开发。这样，软件开发和维护便出现一系列严重问题，即软件危机。这些问题不仅仅限于所开发出的软件成本高且质量得不到保证，还包括如何开发软件，如何维护现有软件等等。与经济危机“商品供过于求”的特点相比，软件危机则是“软件供不应求”。究其原因，一方面是由软件本身的特点引起的，如软件越来越复杂、软件故障难于检测等；另一方面则是因为软件开发和维护以及组织管理不当造成的，如软件开发的个性色彩太浓、软件需求不充分、软件开发过程无序等。

3.软件工程阶段

1968年北大西洋公约组织（NATO）的科学委员会在联邦德国召开的有关软件危机的讨论会上，首次正式提出了“软件工程(Software Engineering)”的概念，其主要思路是：要把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，特别是人类从事计算机硬件研究和开发的经验教训，应用到软件的开发和维护中来。从1968年到现在，软件开发有了很大的进步：软件开发工具和开发环境不断改进；软件工程标准和规范不断完善；面向对象技术、软件质量保证、软件复用技术等逐渐得到人们的认同等等。但到目前为止，软件

危机并未完全摆脱，还需人们对软件工程进行更长久更深入的研究。

1.2 软件工程

软件工程是指导软件开发和维护的工程性学科，它以计算机科学理论及其他相关学科的理论为指导，采用工程化的概念、原理、技术和方法进行软件的开发和维护，把经过时间证明是正确的管理措施和当前能够得到的最好的技术、方法相结合，以期用较少的代价获取高质量的软件。

1.2.1 软件工程的研究内容

软件工程是一门综合性的交叉学科，主要涉及计算机科学、管理科学和数学等。计算机科学侧重于理论研究，其成果可应用于软件工程，而软件工程则强调如何有效地建造一个软件系统。管理科学中的方法和原理可用来进行软件生产的管理，包括费用估算、制订计划等。数学可用来建立软件开发中的各种算法和模型。

软件工程作为一门独立的学科，其主要的研究内容是与软件开发和维护有关的 4个方面的内容：标准和规范、过程和模型、方法和技术、工具和环境，软件工程管理则贯穿于这 4 个方面。

作为工程而言，标准化、规范化可以使各种工作有章可循，进而提高软件的生产效率和软件产品的质量。软件工程标准主要有 5 个层次 国际标准、国家标准、行业标准、企业规范和项目规范，其详细内容将在后面继续讨论。

所有软件开发和维护都由一系列过程所构成，建立在一定的标准和规范的基础上，将方法和技术、工具和环境相结合，以便合理、及时地进行软件开发和维护。软件生存期模型则将软件过程有机结合起来，提供一个结构框架，明确主要活动和任务，忽略次要细节，以利于软件开发和维护的各类人员理解，适应不同的项目，有时也称为过程模型，如瀑布模型、原型模型等。由此看来，过程强调的是具体的活动和任务，模型则突出表现过程的有机结合。

软件开发和维护方法体现了软件开发和维护人员看待系统的立场和观点，即方法论意义上的方法。例如，结构化方法认为系统是由一些结构化的功能相互联系、相互作用而构成；面向对象方法则认为系统是由一些对象的相互联系、相互作用而构成。技术则是方法的具体实现，由若干步骤组成，突出“如何做”，有时也不加区别地称之为方法，即技术方法，如 SA/SD(Structured Analysis/Structured Design) 方法是一种结构化分析与设计技术、OMT(Object Modeling

Technique）方法是一种面对象分析和设计技术等等，读者可以依据有关的上下文来加以区分。除此之外，还有一些与软件开发和维护有关的辅助技术，如软件复杂性分析技术、软件可靠性分析技术等。

工具为软件开发和维护的方法和技术提供了自动或半自动的软件支持，以提高软件生产质量和效率，降低软件开发和维护成本，如编译程序、测试工具等。将各种工具结合起来，连同有关的软硬件便形成软件开发和维护环境，其目的是使工具支持整个软件生存周期，如北大的青鸟系统、中科院软件所的 XYZ 系统等。

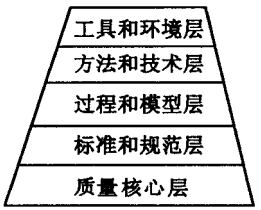

图 1－2　软件工程层次结构

软件工程的四个方面的研究内容构成了以软件质量为核心的层次结构，如图 1－2 所示。全面质量管理和类似的管理技术孕育了不断发展和提高的企业文化，而正是这种企业文化导致了不断成熟的软件工程标准和规范、过程和模型、方法和技术、工具和环境的发展。

1.2.2　软件工程的基本原理

1983 年，美国著名的软件工程专家 Boehm 综合 100 多条关于软件工程的准则和信条，并总结 TRW 公司多年的软件开发经验，提出了软件工程的 7 条基本原理。Boehm 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。

1. 用分阶段的生命周期计划严格管理

统计表明，在不成功的软件项目中有一半左右是由于软件开发和维护计划不周造成的。在软件开发和维护的生命周期中，需要完成许多性质不同的工作。把软件生命周期划分为若干阶段，并相应地制订出切实可行的计划，然后严格按计划对软件的开发和维护工作进行管理，是软件项目成功的关键。

Boehm 认为，在软件的整个生命周期中应该制订六类计划，即：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。

2. 坚持进行阶段评审

统计表明，大部分的软件错误是在编码之前造成的，约占 63％。同时，错误发现得越晚，改正错误付出的代价也就越大。因此，必须坚持严格的阶段评审，以便尽早发现错误。

3. 实行严格的产品控制

由于外部环境的变化，用户需求的改动往往是不可避免的。但在软件开发过程中改变需求需要付出较高的代价，因此不能随意改变需求，要采用严格的产

品控制技术，其中主要是实行基准配置管理。所谓基准配置又称基线配置，它们是经过阶段评审后的软件配置成分，如各阶段产生的文档或程序代码。基准配置管理也称变动控制：一切有关软件修改的建议，特别是涉及到对基准配置修改的建议，必须经过严格的规程进行评审，获得批准后才能进行有关修改。

4.采纳现代程序设计技术

从提出软件工程概念开始，人们都致力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，已经成为当时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化分析和结构化设计技术。实践表明,采用先进的技术既可提高软件开发的效率,又可提高软件维护的效率。

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，从而使软件产品的开发过程比一般产品的开发过程难于评价和管理。因此，应该根据软件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，提高软件开发过程的可见性，从而使所得的结果能够清楚地审查。

6.开发小组的人员应少而精

当开发小组人员数为N时，可能的通信路径有N*(N－1)/2条。显然，随着人数N的增大，人员之间的通信路径急剧增加，造成通信开销的剧增。同时，高素质的人员开发软件的效率比低素质人员开发软件的效率可能高出几倍甚至几十倍，而且错误数明显比低素质人员开发的软件中的错误数少。因此，软件开发小组的人员组成应该少而精。

7.承认不断改进软件工程实践的必要性

Boehm认为：遵循软件工程的前6条原理，就能够较好地实现软件的工程化生产，但不能保证跟得上技术的不断进步。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，而且应不断总结经验，收集软件开发和维护过程中的有关数据，例如：进度和资源耗费数据、出错类型和统计数据等。经验和数据不仅可用于评价新的技术，而且为新技术的发展提供了重要依据。

1.3 软件生存期过程

软件生存期过程规定了获取、供应、开发、操作和维护软件时，要实施的过程、活动和任务。其目的是为各种人员提供一个公共的框架，以便可以使用“相同的语言”在自己的环境中创作和管理软件。1995年制订和公布的国家标准

《GB/T8566－1995 信息技术——软件生存期过程》定义了软件生存期 7个主要过程：管理过程、获取过程、供应过程、开发过程、操作过程、维护过程和支持过程，它们之间的作用和关系如图 1－3所示。

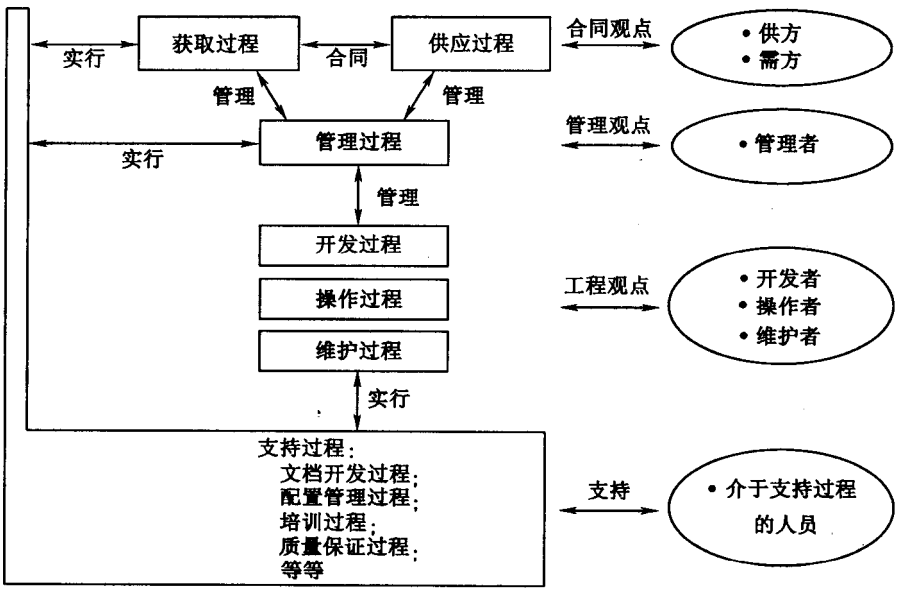

图 1－3 软件生存期过程——作用和关系

1.3.1 获取和供应过程

获取和供应过程从合同的观点分别定义了需方和供方的活动和任务。获取过程从定义软件产品或服务的获取需求开始，接着准备并公布标书、选择供方和管理获取过程，直到系统验收。供应过程的开始方法有两种：一是决定准备一项建议以应答需方的标书；二是就提供一个含有软件的系统（或系统的一个部件、一个产品或一项服务）与需方签订合同或协议。接着就是规定为了管理和保证项目所需的步骤和资源，其中包括制订项目计划和实施计划，直至向需方交付系统、产品或服务。

1.3.2 管理过程

管理过程从管理的观点上定义了一个机构在管理自己的过程中管理者所需的活动和任务。管理过程从对将要采取的过程规定需求开始。接着，管理人员应当制订实施此过程的计划并实施该计划；监督过程的实施；调查、分析和解决执行此过程中发现的问题。最后，评审和评价在过程实施中所完成的产品，保证

产品的完整性和一致性。

1.3.3 开发、操作和维护过程

按照工程的观点，开发者、操作者或维护者分别通过实施开发过程、操作过程和维护过程，生产软件产品或提供服务。开发过程包括系统需求分析、系统设计、软件需求分析、软件体系结构设计、软件的详细设计、软件编码、软件集成、软件鉴定测试、系统集成以及系统鉴定测试和验收等活动。按照合同，软件开发者的责任从软件需求分析开始，以软件鉴定测试终止。但通常软件是作为整个系统的一部分实现的，因而软件开发者可能要参加系统需求分析、系统设计，以从中获得必要的信息。操作过程包括系统操作和对用户的操作支持。当系统由于错误、缺陷、问题或需要改进和修改，从而要修改代码和相关的文档时即进入维护过程。维护过程以系统退役而终止。

1.3.4 支持过程

支持过程用来支持软件项目的生存期过程，有助于项目的成功和提高项目的质量。支持过程包括文档开发过程、配置管理过程、合同要求的评审和审计过程、验证和确认过程、软件质量保证过程、改正过程、培训过程和环境建立过程等8个过程，表 1－1给出了8个过程的有关活动和任务。

表 1－1 支持过程的主要活动和任务

过程名称	有关活动和任务
文档开发过程	计划、设计和开发、编辑、发行和维护文档。
配置管理过程	计划、配置标识、配置控制、配置状态计算、配置审计、软件和文档的储存、处理和交付控制。
合同要求的评审和审计过程	管理评审和技术评审；功能性配置审计、物理配置审计、对进行中的过程审计。
验证和确认过程	需求验证、设计验证、代码验证、集成验证和文档评审；为分析测试结果实施特定测试；确认软件正确地实现了关键性、安全性和保密要求等。
软件质量保证过程	产品保证、过程保证和质量改进。
改正过程	分析、开发、运行、维护或其他过程中出现的问题，提出相应对策，采取改正活动。
培训过程	制订培训计划；编写培训资料；培训计划实施。
环境建立过程	对环境的配置作计划；安装环境；维护环境。

软件生存期过程没有规定一个特定的生存周期模型，各软件开发机构可视其项目的需要选择一种软件生存周期模型，并将软件生存期过程所含过程、活动和任务映射到选定的软件生存周期模型中。

1.4 常用软件生存期模型

软件生存期模型描述了软件项目从需求定义开始，到开发成功后投入使用，在使用的过程中不断增补修订，直到最后停止使用这一期间所进行的各种活动是如何执行的模型。目前，人们提出并实践了许多种生存期模型，如瀑布模型、原型模型、快速应用开发(RAD)模型等。软件开发机构应该综合项目和应用的性质、将要使用的方法和工具等，选择其中合适的模型，并将软件生存期过程映射到选定的模型中进行软件开发和维护。

1.4.1 瀑布模型

瀑布模型也称传统的生命周期模型，将软件生存期的活动和任务规定为依线性顺序联接的若干阶段。瀑布模型最初由 W. Royce 于 1970 年提出，一个典型的瀑布模型如图 1－4所示。

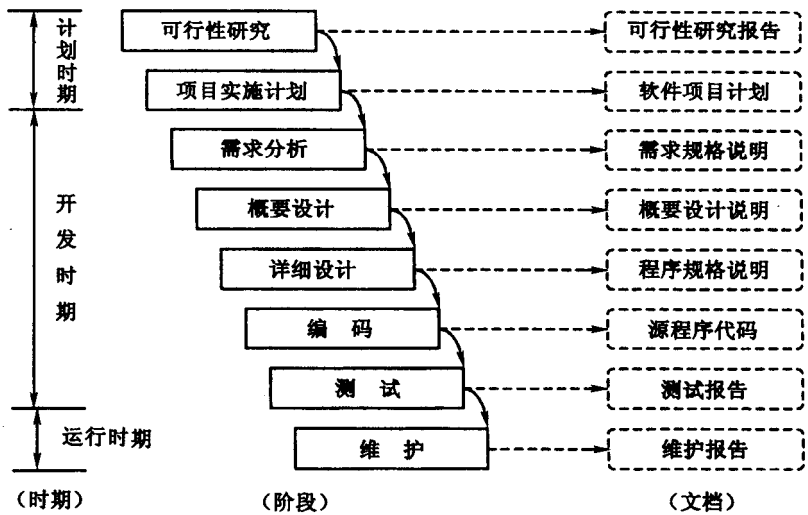

图 1－4 典型的瀑布模型

瀑布模型把软件生存期划分为三个时期：计划、开发和运行。每一个时期又区分为若干阶段，各阶段的工作顺序展开，如同拾级而下的瀑布，每一阶段均以上一阶段经过评审的文档作为开发的基础。

然而，软件开发的实践表明，上述各阶段之间并非完全由自上而下的线性顺序展开。实际软件开发中，每阶段具有以下特征：

- 上一阶段的文档作为本阶段的输入；
- 利用这一输入实施本阶段应完成的内容；
- 给出本阶段的文档作为下一阶段的输入；
- 对本阶段的工作进行评审，若其工作得到确认，则继续进行下一阶段；否则返回前一阶段，甚至更前面的阶段进行返工。

瀑布模型为软件开发和维护提供了一种有效的管理方式。瀑布模型强调严格控制生存期各个阶段来开发和维护软件；每一阶段必须提供相应的文档；这些文档要求必须由软件质量保证（SQA）小组进行严格的评审，以便尽早发现问题,消除隐患,降低成本。

瀑布模型比较适合于功能和性能需求明确的软件项目的开发和维护，如编译系统、数据库管理系统和操作系统等。

实际应用表明，在软件开发前期，用户对系统需求往往只有一个模糊的想法，无法准确表达对系统的全面要求；在开发过程中，用户看不见系统，而只有在交付使用时系统才能和用户见面。这些问题会使得开发出的软件并不是用户真正需要的软件，从而导致大量返工。随着软件项目规模的日益庞大，瀑布模型的这种不够灵活会更加明显，最坏时可能导致软件项目以失败而告终。为了克服瀑布模型的不足，现已提出了许多其他模型。

1.4.2 原型模型

原型模型的主要思想是：快速建立一个反映用户主要需求的原型，供开发人员和用户进行交流，用户判断哪些方面需要改进，然后开发人员改进原型，如此反复，直到原型符合用户要求为止，最终建立完全符合用户要求的新系统。图 1－5 给出了快速原型模型示意图。

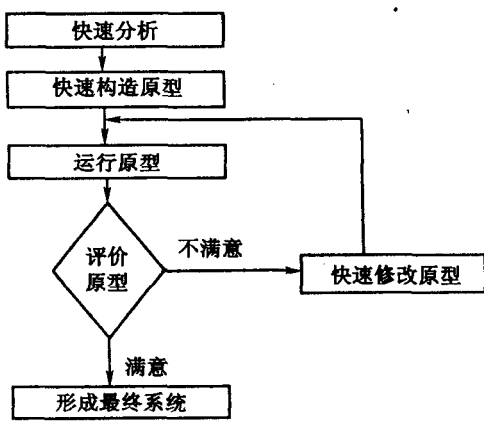

图 1－5 快速原型模型示意图

原型模型通过向用户提供原型获取用户的反馈，使开发出的软件能够真正反映用户的需求。同时，原型模型采用逐步求精方法完善原型，使得原型能够

“快速”开发，避免了像瀑布模型一样在冗长的开发过程中难以对用户的反馈作出快速响应。相对瀑布模型而言，原型模型更符合人们开发软件的习惯，是目前较流行的一种实用软件生存期模型。

在快速建立系统原型时，往往尽量使用能缩短开发周期的语言和工具，实现系统的主要功能以及系统的重要接口，对系统的细节，如异常处理、系统的性能等推迟考虑。因此，不宜利用原型系统作为最终产品。除少数简单的事务系统外，大多数原型可能都会废弃不用，仅把建立原型的过程当作帮助定义软件需求的一种手段。由此，原型模型的“快速”特点对最终系统是不适用的，但要向用户阐述清楚，对最终产品像原型一样快速修改可能会比较困难。因而，采用原型模型开发系统，用户和开发者必须达成一致：原型被建造仅仅是用来定义需求，之后便部分或全部抛弃，最终的软件要在充分考虑了质量和可维护性等方面之后才被开发。

1.4.3 RAD 模型

快速应用开发(Rapid Application Development，RAD)模型同瀑布模型类似，也是一个线性顺序的软件开发模型。但该模型强调极短的开发周期，通过使用基于组件的建造方法获得快速开发。如果软件项目需求清楚且约束了项目范围，RAD 模型能够在很短时间内（如 2～3 个月）创建出“功能完善的系统”。图 1－6 给出了 RAD 模型。

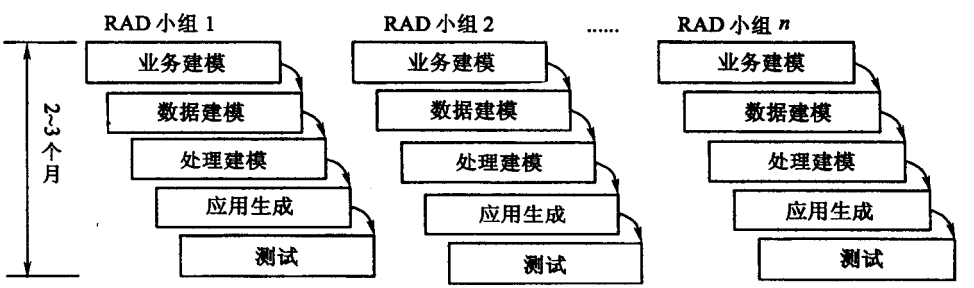

图 1－6 RAD 模型

在图 1－6 中，每个 RAD 小组通过五个顺序阶段进行软件开发：业务建模、数据建模、处理建模、应用生成和测试。业务建模是指对业务活动中的信息流进行建模，以回答如下问题：什么信息驱动业务过程？生成了哪些信息？谁生成该信息？该信息流往何处以及谁处理这些信息？数据建模则进一步精化业务建模阶段定义的一部分信息，形成一组支持该业务所需的数据对象。标识出每个对象的属性，并定义这些对象间的关系。处理建模则从数据建模中得到的数据对象出发，根据每

一个要完成的业务功能的需要，创建处理描述以便增加、修改、获取或删除某个数据对象。应用生成阶段则通过复用已有的程序组件或创建可复用的组件，使用自动化工具辅助建造软件。由于 RAD 在应用生成阶段强调复用，许多程序组件已经经过测试，因此，测试阶段主要测试新的组件以及所有接口。

RAD 模型主要用于信息系统应用软件的开发。如果一个商业应用能够分成若干功能，使得其中每一个主要功能均可在 2～3 个月的时间内通过 RAD 的四个阶段完成，则可以应用 RAD 模型开发。将每一个主要功能由单独的 RAD 小组来实现，最后集成一个系统。但是，使用这种模型也有它的局限性：对于大型项目，RAD 需要足够的人力以创建足够的 RAD 小组，RAD 要求承担必要的快速活动的开发者和用户在很短的时间内完成一部分系统，任何一方没有完成约定，都会导致 RAD 项目失败；同时，技术风险很高的情况，如新软件要求与已存在的程序有高可互操作性时，或系统难以被适当地划分为若干功能等情况，则不适合采用 RAD 模型。

1.4.4 增量模型

前面叙述的三种生存期模型，均是一次性地将整个系统提交给用户。瀑布模型是假设当线性阶段完成之后就能交付一个完整的系统。原型模型主要用来帮助开发者获取用户需求，待需求稳定后再开发最终系统提供给用户。 RAD 模型则先将系统主要功能分给若干 RAD 小组开发，然后集成起来形成最终系统提交给用户。但是业务和产品需求的变化、市场竞争和商业压力等等，使得软件开发者必须渐进地开发，以逐步增加软件产品的方式构造软件。

增量模型是一种迭代的演化模型，它结合了瀑布模型和原型模型的思想。例如，使用增量模型开发文档编辑软件，第一个增量可能是发布基本的文件管理、编辑和文档生成功能；第二个增量可能是发布拼写和文法检查功能；第三个增量则完成表格、图形等功能等等。增量模型的每个增量开发过程是瀑布式的，但每一个增量又像原型模型一样是在前一轮增量的基础上进行的。当每个增量交付用户使用（或进行更细的复审）后，开发者根据用户使用和评估的结果对产品进行修改，以满足用户需要，同时发布一些新增的特点和功能。这样不断的重复，直至产生最终的完善产品。图 1－7 给出了一种增量模型的示意图。

增量模型有许多优点。早期的增量开发可以由较少的人员实现，以后的增量开发则可以根据需要补充人员。同时，增量开发能够有计划地管理技术风险。例如，系统的某个重要部分需要使用正在开发但发布时间尚未确定的新产品，则可计划在早期的增量中避免使用该新产品。这样，用户可以先期得到部分可以使用的功能，避免因新产品而使系统的问世时间延迟。另外，由于软件产品是逐

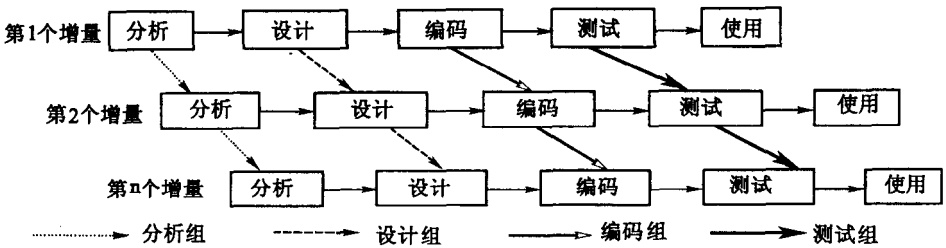

图 1－7　增量模型示意图

渐提供给用户的，这样能够减少一个全新软件产品对用户所带来的心理上的影响。从经济上来看，增量式地交付产品不需要大的资金支出。同时，客户可以在任何时候停止产品的开发，而不必完成整个产品来得到投资上的回报。

增量模型面临的一个困难是：每次增加的增量必须能合并到已有的结构中。如果产品整体结构设计不当，难以为其增加新的增量，即产品是不可扩充的，则增量模型开发会导致一个不满意的产品，甚至会导致项目失败。

1.4.5　螺旋模型

螺旋模型最早由 Boehm 提出，如图 1－8 所示。螺旋模型结合了诸多模型的特征，并加上了风险分析。螺旋模型中的每一圈对应一个阶段。每个阶段从左上的第一象限开始，确定该阶段的目标、实现这些目标的方案以及对这些方案的限制条件。接下来，从风险分析的角度分析这些目标、方案和限制条件，试图排除每个潜在的风险，必要时通过建造原型来排除。如果某些重大风险不能排除，则项目应立即停止。对于某些风险，可以继续进行项目的开发，但这种情况占的比例必须相当小。如果所有风险成功消除，则进入工程开发阶段，螺旋模型的这个象限对应着瀑布模型的各个阶段。最后对工程开发阶段的成果进行评估，并计划下一阶段。多数情况下，沿螺旋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。

螺旋模型适合于大型软件的开发。风险分析使得用户和开发者能够更好地理解和对待每一个阶段的风险。同时，在每一阶段均可应用原型来消除或降低风险的机制是一种比较适用的方法。如果应用得当，能够在风险变成危害之前降低它的影响。螺旋模型既保持了传统生命周期模型中系统的阶段性方法，又将迭代演化的思想吸收到模型中，汇集了诸多模型的优点，更加真实地反映了软件开发的实际情况。特别是在螺旋模型中，维护和开发之间没有什么本质上的差别，维护只是另一个螺旋循环开始而已。当然，螺旋模型在每一阶段开始强调可选方案和限制条件的确定，使得螺旋模型支持对已有软件的复用，并能把软件

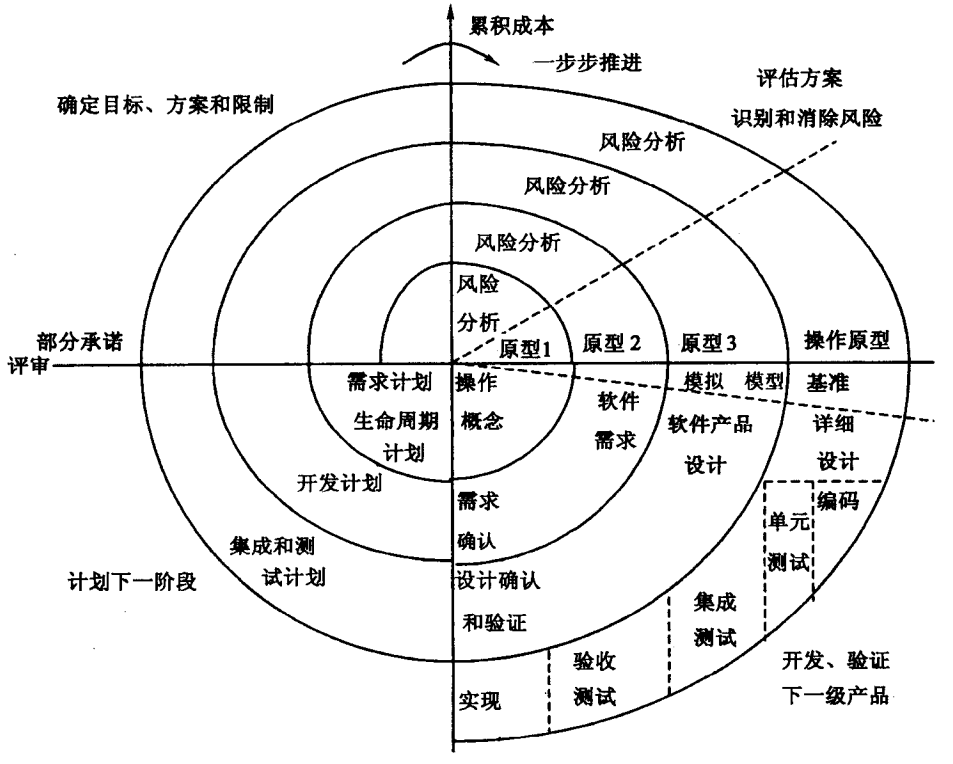

图 1－8 螺旋模型

质量作为特定的目标结合在其中。

螺旋模型是风险驱动的，这对螺旋模型的使用带来一些限制。首先，要求软件开发人员善长风险分析，否则真正的风险是：重大风险没有被开发人员识别，使其造成重大损失。风险分析也使得这种模型不适应于签合同的软件开发，因为风险分析会导致项目终止，而终止合同会导致违约诉讼。另外，对于小项目，风险分析的成本可能与整个项目的成本相当，这样对于规模较小的项目利用螺旋模型开发会导致成本过大。

1.5 软件工程标准

软件工程标准是软件工程发展的必然产物，也是人们对计算机软件认识深化的结果。在软件开发的过程中，需要许多不同层次、不同分工的人员的协作。

为便于人员之间的信息交流，须制订相应的标准，规范软件开发过程和产品。同时，遵照标准，可以提高软件产品的质量和效率，降低软件产品的成本。

1.5.1 软件工程标准类型

软件工程标准类型是多方面的，且标准的范围和内容与软件工程有关方面的特性相关。GB/T 15538—1995 采用的分类法由标准划分、软件工程划分和这两种划分的表示关系组成，用二维表格来描述，标准划分确定了标准的作用，软件工程划分确定了与标准有关的软件工程方面的特性。

GB/T 15538—1995 的标准划分有四种类型：过程标准、产品标准、行业标准和记法标准。图 1－9 给出了按类型对标准的完整划分。过程标准同开发一项产品或从事一项服务的一系列活动或操作有关，这些活动或操作使用一些方法、工具和技术。产品标准涉及事物的格式和内容。软件开发和维护活动的文档化结果就是软件产品，它给出了进一步工作的基础。软件工程的行业标准涉及软件工程的所有方面，如软件过程能力成熟度模型（Capability Maturity Model，CMM）认证。记法标准论述了在软件工程行业范围内，以惟一的方式进行交流的方法。

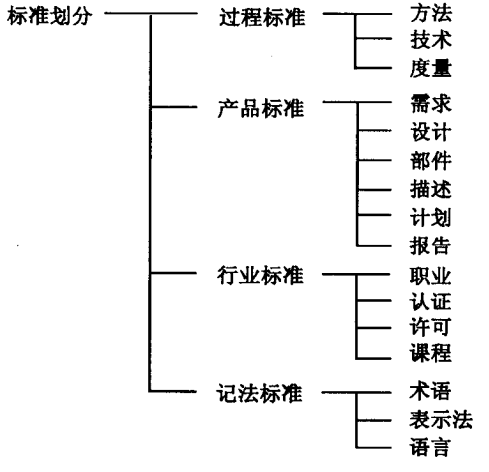

图 1－9 GB/T 15538—1995 标准划分

GB/T 15538—1995 的软件工程划分包括两部分：任务功能和软件生存周期。使用这两部分以便比较、判断、评价和确定软件工程标准的范围和内容。

任务功能可划分为三部分：产品工程功能、验证与确认功能、技术管理功能。产品工程功能是定义、产生和支持最终软件产品所必须的过程，包括：需求分析、设计、编码、集成、转换、排错、调试、产品支持和软件维护。验证与确认功能是检查产品质量的技术活动，包括：评审和审计、产品分析和测试。技术管理功能是构造和控制产品工程功能的有关过程，包括：过程管理、产品管理和资源管理。这三部分包括的活动不是集中在单个生存期阶段中，而是并行进行的产生、检查和控制的主要活动。

1.5.2 软件工程标准层次

根据软件工程标准制定的机构与适用范围的不同，软件工程标准可分为 5 个由大到小、由普通到特殊的层次，即国际标准、国家标准、行业标准、企业（机构）规范和项目（课题）规范等。

1. 国际标准

由国际联合机构制定和公布，提供给各国参考的标准。

国际标准化组织（International Standards Organization，ISO）是一个有着广泛的代表性和权威性的国际机构，它所公布的标准有较大的影响。如 ISO 8631－86 Information processing—program constructs and conventions for their representation（《信息处理——程序构造及其表示法的约定》）、ISO 9000 族标准（质量体系方面）等等。

2. 国家标准

由政府或国家级的机构制定或批准，适合于全国范围的标准。例如：

• GB——中华人民共和国国家质量技术监督局是中国的最高标准化机构，其所公布的标准简称"国标"。

例如 GB/T 8566－1995（信息技术、软件生存期过程）、GB/T 11457－1995（软件工程术语）、GB 13502－1992（信息处理程序、构造及其表示的约定）等。

• ANSI（American National Standards Institute）——美国国家标准协会。

• BS（British Standards）——英国国家标准。

• DIN（Detaches Institute Fur Normung）—— 德国标准协会。

• JIS（Japanese Industrial Standard）——日本工业标准。

3. 行业标准

由行业机构、学术团体或国防机构制定的适合于某个行业的标准。例如：

• GJB—— 中华人民共和国国家军用标准。如 GJB 437－88（军用软件开发规范）、GJB 438－88（军用软件文档编制规范）等。

• IEEE（Institute of Electrical and Electronics Engineers）——美国电气与电子工程师学会。由于 IEEE 通过的标准经常报请 ANSI 审批，使之具有国家标准的性质。因此，IEEE 公布的标准常冠有 ANSI 的字头，如 ANSI/IEEE 828－1983（软件配置管理计划标准）等。

4. 企业规范

大型企业或公司由于软件工程工作的需要而制定的适用于本部门的规范。

例如，美国 IBM 公司通用产品部 1984 年制定的《程序设计开发指南》，仅供该公司内部使用。