第 1 章　软件工程概述

1.1　软件工程的产生和发展

软件工程（Software Engineering）是在克服 20 世纪 60 年代末国际上所出现的“软件危机”的过程中逐渐形成与发展的。自 1968 年北大西洋公约组织（NATO）所举行的软件可靠性学术会议上为克服软件危机，正式提出“软件工程”的概念以来，软件工程在理论和实践两方面都取得了长足的进步。

软件工程是一门指导计算机软件系统开发和维护的工程学科，是一门新兴的边缘学科，它涉及计算机科学、管理学、数学等多个学科，其研究范围广，不仅包括软件系统的开发方法和技术、管理技术，还包括软件工具、环境及软件开发的规范。

软件是信息化的核心。国民经济、国防建设、社会发展及人民生活都离不开软件。软件产业关系到国家经济发展和文化安全，体现了一个国家的综合实力，是决定 21 世纪国际竞争地位的战略性产业。尤其是随着互联网技术的迅速发展，软件工程对促进信息产业发展和信息化建设的作用凸现。

因此，大力推广应用软件工程的开发技术及管理技术，提高软件工程的应用水平，对促进我国软件产业与国际接轨，推动软件产业的迅速发展将起着十分重要的作用。

1.1.1　软件危机与软件工程

1．软件危机

20 世纪 60 年代末期，随着软件的规模越来越大，复杂度不断增加，软件需求量也不断增大，而当时生产作坊式的软件开发模式及技术已不能满足软件发展的需要。

软件开发过程是一种高密集度的脑力劳动，需要投入大量的人力、物力和财力；由于软件开发的模式及技术不能适应软件发展的需要，致使大量质量低劣的软件产品涌向市场，有的甚至在开发过程中就夭折了。国外在开发一些大型软件系统时，遇到了许多困难，有的系统最终彻底失败了；有的系统则比原计划推迟了好多年，而费用大大超过了预算；或者系统功能不符合用户的需求；也无法进行修改维护。典型的例子有：

IBM 公司的 OS/360，共约 100 万条指令，花费了 5000 个人年，经费达数亿美元，而结果却令人沮丧，错误多达 2000 个以上，系统根本无法正常运行。OS/360 系统的负责人 Brooks 这样描述开发过程的困难和混乱：“像巨兽在泥潭中做垂死挣扎，挣扎得越猛，泥浆就沾得越多，最后没有一个野兽能够逃脱淹没在泥潭中的命运……”。

1967 年苏联“联盟一号”载人宇宙飞船，由于其软件忽略一个小数点的错误，导致返航时打不开降落伞，当进入大气层时因摩擦力太大而烧毁，造成机毁人亡的巨大损失。

还有，可以称为 20 世纪世界上最精心设计，并花费了巨额投资的美国阿波罗登月飞行计划的软件，也仍然没有避免出错。例如，阿波罗 8 号太空飞船由于计算机软件的一个错误，造成存储器的一部分信息丢失；阿波罗 14 号在飞行的 10 天中，出现了 18 个软件错误。

2．软件危机的表现

软件危机，反映在软件可靠性没有保障、软件维护工作量大、费用不断上升、进度无法预测、成本增长无法控制、程序人员无限度地增加等各个方面，以至于形成人们难以控制软件开发的局面。

软件危机主要表现在两个方面：

① 软件产品质量低劣，甚至在开发过程中就夭折。

② 软件生产效率低，不能满足需要。

3．软件工程的概念

软件危机所造成的严重后果已使世界各国的软件产业危机四伏，面临崩溃，克服软件危机刻不容缓。

1968 年，在北大西洋公约组织所召开的可靠性会议上，首次提出了“软件工程”的概念，即借鉴工程化的方法来开发软件。自该会议以来，世界各国的软件工作者为克服软件危机进行了许多开创性的工作，在软件工程的理论研究和工程实践两个方面都取得了长足的进步，缓解了软件危机。但距离彻底克服软件危机这个软件工程的最终目标，任重道远，还需要软件工作者付出长期艰苦的努力。

从“软件工程”的概念提出至今，软件工程的发展已经历了四个重要阶段：

（1）第一代软件工程

20 世纪 60 年代末所出现的“软件危机”，其表现为软件生产效率低，大量质量低劣的软件涌入市场，甚至在软件开发过程中夭折，使软件产业濒临瘫痪。

为克服“软件危机”，在著名的 NATO 软件可靠性会议上第一次提出了“软件工程”的术语以来，将软件开发纳入了工程化的轨道，基本形成了软件工程的概念、框架、技术和方法。这一阶段又称为传统的软件工程。

（2）第二代软件工程

20 世纪 80 年代中期，以 Smalltalk 为代表的面向对象的程序设计语言相继推出，面向对象的方法与技术得到发展；从 20 世纪 90 年代起，研究的重点从程序设计语言逐渐转移到面向对象的分析与设计，演化为一种完整的软件开发方法和系统的技术体系。20 世纪 90 年代以来，出现了许多面向对象的开发方法的流派，面向对象的方法逐渐成为软件开发的主流方法。所以这一阶段又称为对象工程。

（3）第三代软件工程

随着软件规模和复杂度的不断增大，开发人员也随之增多，开发周期也相应延长，加之软件是知识密集型的逻辑思维产品，这些都增加了软件工程管理的难度。人们在软件开发的实践过程中认识到：提高软件生产效率，保证软件质量的关键是对“软件过程”的控制和管理，即是对软件开发和维护中的管理和支持能力。提出了对软件项目管理的计划、组织、成本估算、质量保证、软件配置管理等技术与策略，逐步形成了软件过程工程。

（4）第四代软件工程

20 世纪 90 年代起至今，基于组件（Component）的开发方法取得重要进展，软件系统的开发可通过使用现存的可复用组件组装完成，而无须从头开始构造，以此达到提高效率和质量，降低成本的目的。软件复用技术及组件技术的发展，对克服软件危机提供了一条有效途径，将这一阶段称为组件工程。

1.1.2 软件工程的定义及基本原则

1. 软件工程的定义

自 1968 年提出软件工程这个术语以来，软件工程一直以来都缺乏统一的定义，很多学者、组织机构都分别给出了自己认可的定义。

例如，1983 年，IEEE（国际电气与电子工程师协会）所下的定义是：软件工程是开发、运行、维护和修复软件的系统方法。1990 年，IEEE 又将定义更改为：对软件开发、运作、维护的系统化的、有规范的、可定量的方法之应用，即是对软件的工程化应用。

ISO 9000 对软件工程过程的定义是：软件工程过程是输入转化为输出的一组彼此相关的资源和活动。

BarryBoehm 则定义为：运用现代科学技术知识来设计并构造计算机程序及为开发、运行和维护这些程序所必需的相关文件资料。

对于软件工程的各种各样的定义，它们的基本思想都是强调在软件开发过程中应用工程化原则的重要性。

从软件工程的定义可见，软件工程是一门指导软件开发的工程学科，它以计算机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，把经实践证明的科学的管理措施与最先进的技术方法结合起来。即软件工程研究的目标是“以较少的投资获取高质量的软件”。

2. 软件工程的基本原则

软件工程的基本原则是随着软件工程的发展而变化的。过去软件工程的基本原则是抽象、模块化、清晰的结构、精确的设计规格说明。但今天的认识已经发生了很大的变化，软件工程 4 条基本原则是：

① 必须认识软件需求的变动性，以便采取适当措施来保证产品能最好地满足用户要求。在软件设计中，通常要考虑模块化、抽象与信息隐蔽、局部化、一致性等原则。

② 稳妥的设计方法将大大方便软件开发，以达到软件工程的目标。软件工具与环境对软件设计的支持来说，颇为重要。

③ 软件工程项目的质量与经济开销取决于对它所提出的支撑质量与效用。

④ 只有在强调对软件过程进行有效管理的情况下，才能实现有效的软件工程。

1.1.3 软件工程研究的内容

软件工程是一门新兴的边缘学科，涉及的学科多，研究的范围广。归结起来软件工程研究的主要内容有以下几方面：方法与技术、工具及环境、管理技术、标准与规范。

① 软件开发方法，主要讨论软件开发的各种方法及其工作模型，它包括多方面的任务，如软件系统需求分析、总体设计，以及如何构建良好的软件结构、数据结构及算法设计等，包括具体实现的技术。

② 软件工具为软件工程方法提供支持，研究计算机辅助软件工程，建立软件工程环境。

③ 软件工程管理，是指对软件工程全过程的控制和管理，包括计划安排、成本估算、项目管理、软件质量管理等。

④ 软件工程标准化与规范化，使得各项工作有章可循，以保证软件生产效率和软件质量

的提高。软件工程标准可分为 4 个层次：国际标准、行业标准、企业规范和项目规范。

此外，按照 ACM 和 IEEE-CS 发布的软件工程知识体系（SWEBOK）定义的软件工程学科的内涵，软件工程研究的内容由 10 个知识域构成。

（1）软件需求（Software Requirements）。软件需求涉及需求抽取、需求分析、建立需求规格说明和确认等活动，还涉及建模、经济与时间可行性分析。

（2）软件设计（Software Design）。设计是软件工程最核心的内容。其主要活动有软件体系结构设计、软件详细设计。涉及软件体系结构、组件、接口，以及系统或组件的其他特征，还涉及软件设计质量分析和评估、软件设计的符号、软件设计策略和方法等。

（3）软件构造（Software Construction）。通过编码、单元测试、集成测试、调试、确认等活动，生成可用的、符合设计功能的软件。并要求控制和降低程序复杂性。

（4）软件测试（Software Testing）。测试是软件生存周期的重要部分，涉及测试的标准、测试技术、测试度量和测试过程。

（5）软件维护（Software Maintenance）。软件产品交付后，需要改正软件的缺陷，提高软件性能或其他属性，使软件产品适应新的环境。软件维护是软件进化的继续。基于服务的软件维护越来越受到重视。

（6）软件配置管理（Software Configuration Management）。为了系统地控制配置变更，维护整个系统生命周期中配置的一致性和可追踪性，必须按时间管理软件的不同配置，包括配置管理过程的管理、软件配置鉴别、配置管理控制、配置管理状态记录、配置管理审计、软件发布和交付管理等。

（7）软件工程管理（Software Engineering Management）。运用管理活动，如计划、协调、度量、监控、控制和报告，确保软件开发和维护是系统的、规范的、可度量的。它涉及基础设施管理、项目管理、度量和控制计划三个层次。

（8）软件工程过程（Software Engineering Process）。软件工程过程关注软件过程的定义、实现、评估、测量、管理、变更、改进，以及过程和产品的度量。

（9）软件工程工具和方法（Software Engineering Tools and Methods）。软件开发工具是以计算机为基础辅助软件生存周期过程的。软件工具的种类很多，如：需求工具、设计工具、构造工具、测试工具、维护工具、配置管理工具、工程管理工具、工程过程工具、软件质量工具等。

软件工程方法支持软件工程活动，典型的有结构化方法、面向数据方法、面向对象方法、原型化方法及基于数学的形式化方法等。

（10）软件质量（Software Quality）。软件质量管理贯穿整个软件生存周期，涉及软件质量需求、软件质量度量、软件属性检测、软件质量管理技术和过程等。

必须要强调的是，随着人们对软件系统研究的逐渐深入，软件工程所研究的内容也在不断更新和发展。

1.2 软件与软件过程

软件工程是在软件生产中采用工程化的方法，并采用一系列科学的、现代化的方法和技术来开发软件的。这种工程化的思想贯穿软件开发和维护的全过程。

为了进一步学习有关软件工程的方法和技术，先介绍软件、软件生存期及软件工程过程这几个重要的概念。

1.2.1 软件的概念和特点

1. 软件及其特点

“软件就是程序，开发软件就是编写程序”是一个错误观点，这种错误观点的长期存在，影响了软件工程的正常发展。

事实上，正如 Boehm 指出的：软件是程序，以及开发、使用和维护程序所需的所有文档。它是由应用程序、系统程序、面向用户的文档及面向开发者的文档四部分构成的。

软件的特点如下。

① 软件是一种逻辑实体，不是具体的物理实体。

② 软件产品的生产主要是研制过程。

③ 软件具有“复杂性”，其开发和运行常受到计算机系统的限制。

④ 软件成本昂贵，其开发方式目前尚未完全摆脱手工生产方式。

⑤ 软件不存在磨损和老化问题，但存在退化问题。

图 1.1 是硬件失效率的“U 形”曲线（浴盆曲线），说明硬件随着使用时间的增加，失效率急剧上升。

图 1.2 所描述的软件失效率曲线，它没有“U 形”曲线的右半翼，表明软件随着使用时间的增加，失效率降低；因为软件不存在磨损和老化问题，但存在退化问题。

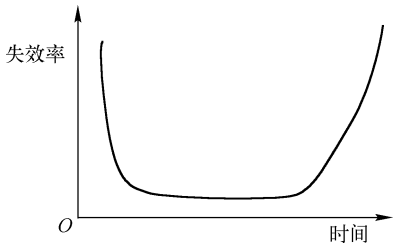

图 1.1 硬件失效率曲线

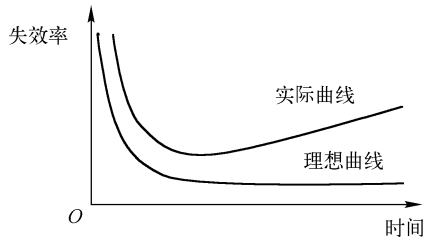

图 1.2 软件失效率曲线

2. 软件生存期

软件生存期（Life Cycle），又称生命周期（SDLD），是指一个从用户需求开始，经过开发、交付使用，在使用中不断地增补修订，直至软件报废的全过程。

软件生命周期分为以下阶段：

① 可行性研究和项目开发计划。该阶段必须要回答的问题是“软件系统要解决的问题是什么”。

② 需求分析。该阶段的任务是，通过分析准确地确定“软件系统必须做什么”，即软件系统必须具备哪些功能。

③ 概要设计。也称总体设计。主要任务是确定软件体系结构，划分子系统模块及确定模块之间的关系。并确定系统的数据结构和进行界面设计。

④ 详细设计。即对每个模块完成的功能、算法进行具体描述，要把功能描述变为精确的、结构化的过程描述。

⑤ 软件构造。该阶段把每个模块的控制结构转换成计算机可接受的程序代码，即编写以某特定程序设计语言表示的“源代码”。

⑥ 测试。是保证软件质量的重要手段，其主要方式是在设计测试用例的基础上检验软件

的各个组成部分。测试分为模块测试、组装测试、确认测试等。

⑦ 维护。软件维护是软件生存期中时间最长的阶段。已交付的软件投入正式使用后，便进入软件维护阶段，它可以持续几年甚至几十年。

特别要指出的是：实际的软件开发过程，是一个充满迭代和反复的过程，上述阶段①到⑥，通常会相互重叠，反复进行，才可能完成软件的开发。

1.2.2 软件工程过程及产品

软件工程过程是指在软件工具的支持下，所进行的一系列软件工程活动。通常包括以下 4 类基本过程：

① 软件规格说明：规定软件的功能及其运行环境。

② 软件开发：产生满足规格说明的软件。

③ 软件确认：确认软件能够完成客户提出的要求。

④ 软件演进：为满足客户的变更要求，软件必须在使用的过程中演进。

软件工程过程具有可理解性、可见性（过程的进展和结果可见）、可靠性、可支持性（易于使用 CASE 工具支持）、可维护性、可接受性（为软件工程师接受）、开发效率和健壮性（抵御外部意外错误的能力）等特性。

软件工程有方法、工具和过程三个要素。软件工程方法研究软件开发“如何做”；软件工具是研究支撑软件开发方法的工具，为方法的运用提供自动或者半自动的支撑环境。软件工具的集成环境，又称为计算机辅助软件工程（Computer Aided Software Engineering，CASE）；软件工程过程则是指将软件工程方法与软件工具相结合，实现合理、及时地进行软件开发的目的，为开发高质量软件规定各项任务的工作步骤。如图 1.3 所示，在软件工程的三要素中，软件过程将人员、方法与规范、工具和管理有机结合，形成一个能有效控制软件开发质量的运行机制。

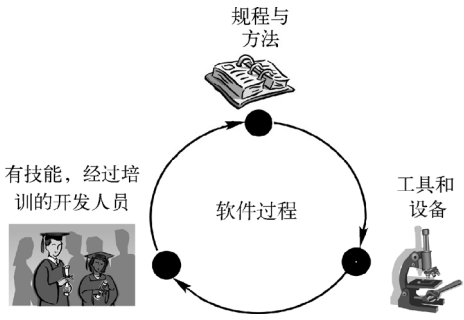

图 1.3 软件工程过程

1.3 软件过程模型

软件过程模型也称为软件生存期模型或软件开发模型，是描述软件过程中各种活动如何执行的模型。它确立了软件开发和演绎中各阶段的次序限制以及各阶段活动的准则，确立开发过程所遵守的规定和限制，便于各种活动的协调以及各种人员的有效通信，有利于活动重用和活动管理。为了描述软件生存期的活动，提出了多种生存期模型，如瀑布模型、循环模型、螺旋模型、喷泉模型、智能模型等。

目前常见的软件过程模型如下。

1.3.1 瀑布模型

瀑布模型是经典的软件开发模型，是 1970 年由 W.Royce 提出的最早的软件开发模型。如图 1.4 所示，瀑布模型将软件开发活动中的各项活动规定为依线性顺序连接的若干阶段，形如瀑布流水，最终得到软件系统或软件产品。换句话说，它将软件开发过程划分成若干个互相区

别而又彼此联系的阶段，每个阶段中的工作都以上一个阶段工作的结果为依据，同时作为下一个阶段的工作基础。每个阶段的任务完成之后，产生相应的文档。该模型适合于需求很明确的软件项目开发。

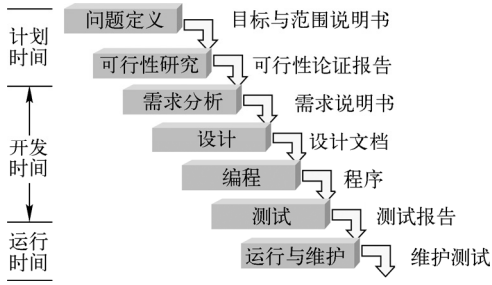

图 1.4　瀑布模型

在软件工程的第一阶段，瀑布模型得到了广泛的应用，它简单易用，在消除非结构化软件，降低软件的复杂性，促进软件开发工程化方面起了很大的作用。但在软件开发实践中也逐渐暴露出它的缺点。由于瀑布模型是一种理想的线性开发模式，它将一个充满回溯的软件开发过程硬性分割为几个阶段，无法解决软件需求不明确或者变动的问题。这些缺点对软件开发带来了严重影响，由于需求不明确，会导致开发的软件不符合用户的需求而夭折。

1.3.2　增量模型

增量模型是一种非整体开发的模型。根据增量的方式和形式的不同，分为基于瀑布模型的渐增模型和基于原型的快速原型模型。一般的增量模型如图 1.5 所示。该模型具有较大的灵活性，适合于软件需求不明确、设计方案有一定风险的软件项目。

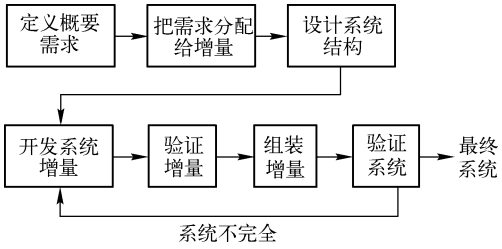

图 1.5　增量模型

增量模型和瀑布模型之间的本质区别是：瀑布模型属于整体开发模型，它规定在开始下一个阶段的工作之前，必须完成前一阶段的所有细节。而增量模型属于非整体开发模型，它推迟某些阶段或所有阶段中的细节，从而较早地产生工作软件。

1.3.3　螺旋模型

对于大型软件，只开发一个原型往往达不到要求。螺旋模型将瀑布模型和增量模型结合起来，并加入了风险分析。它是由 TRW 公司的 B.Boehm 于 1988 年提出的。该模型将开发过程划分为制定计划、风险分析、实施工程和客户评估 4 类活动。如图 1.6 所示，沿着螺旋线每转一圈，表示开发出一个更完善的新的软件版本。如果开发风险过大，开发机构和客户无法接受，项目有可能就此中止；多数情况下，会沿着螺旋线继续下去，自内向外逐步延伸，最终得到满意的软件产品。

螺旋模型将开发过程分为几个螺旋周期，每个螺旋周期可分为 4 个工作步骤：

① 制定计划：确定目标、方案和限制条件；

② 风险分析：评估方案、标识风险和解决风险；

③ 实施工程：开发确认产品；

④ 客户评估：计划下一周期工作。

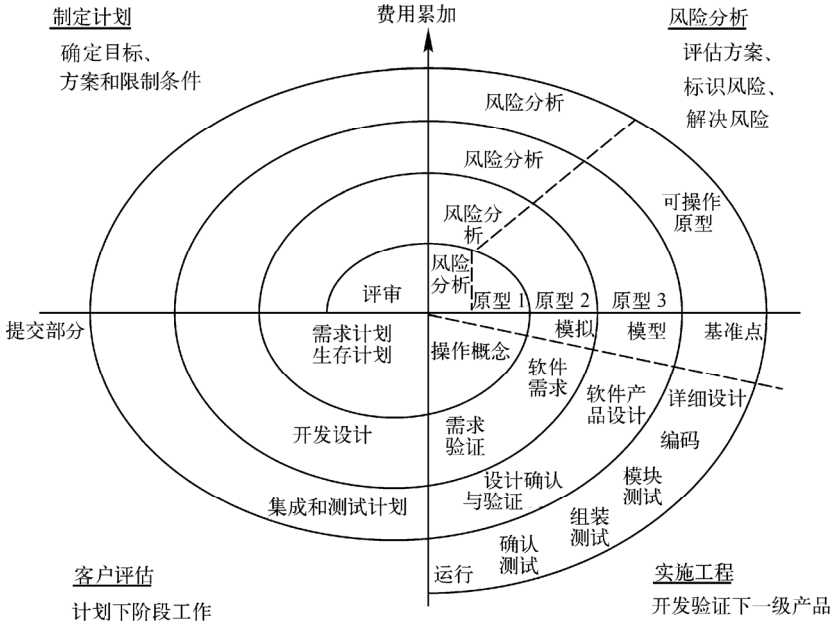

图 1.6　螺旋模型

1.3.4　喷泉模型

喷泉模型是由 B.H.Sollers 和 J.M.Edwards 于 1990 年提出的一种新的开发模型，主要用于采用对象技术的软件开发项目。它克服了瀑布模型不支持软件重用和多项开发活动集成的局限性。喷泉模型使开发过程具有迭代性和无间隙性。软件的某个部分常常被重复工作多次，相关对象在每次迭代中随之加入渐进的软件成分，即为迭代的特性；而分析和设计活动等各项活动之间没有明显的边界，即为无间隙的特性。

喷泉模型以面向对象的软件开发方法为基础，以用户需求作为喷泉模型的源泉。如图 1.7 所示，喷泉模型有如下特点：

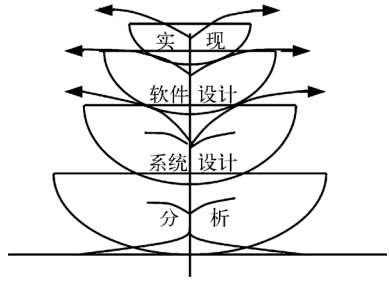

图 1.7　喷泉模型

① 喷泉模型规定软件开发过程有 4 个阶段，即分析、系统设计、软件设计和实现。

② 喷泉模型的各阶段相互重叠，它反映了软件过程并行性的特点。

③ 喷泉模型以分析为基础，资源消耗成塔形，在分析阶段消耗的资源最多。

④ 喷泉模型反映了软件过程迭代性的自然特性，从高层返回低层无资源消耗。

⑤ 喷泉模型强调增量开发，它依据分析一点，设计一点的原则，并不要求一个阶段的彻底完成，整个过程是一个迭代的逐步提炼的过程。

⑥ 喷泉模型是对象驱动的过程，对象是所有活动作用的实体，也是项目管理的基本内容。

⑦ 喷泉模型在实现时，由于活动不同，可分为系统实现和对象实现，这既反映了全系统的开发过程，也反映了对象族的开发和重用过程。

1.3.5　原型模型

原型是软件开发过程中，软件的一个早期可运行的版本，它反映了软件系统的部分重要特

性。原型模型反映了快速建立软件原型的过程。如图 1.8 所示，它是一个循环的模型，通常分为以下 4 步：

① 快速分析。快速确定软件系统的基本要求，确定原型所要体现的主要特征（界面、总体结构、功能、性能）。

② 构造原型。在快速分析的基础上，根据系统的基本规格说明，忽略细节，只考虑主要特征，快速构造一个可运行的系统。

③ 运行和评价原型。用户试用原型并与开发者之间频繁交流，发现问题，目的是验证原型的正确性。

④ 修改与改进。根据所发现的问题，对原型进行修改、增删和完善。

这 4 步按箭头顺序反复执行，直到用户对生成的原型评价满意为止。

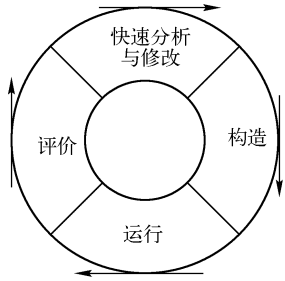

图 1.8　速成原型模型

1.3.6　智能模型

智能模型也称为基于知识的软件开发模型，是知识工程与软件工程在开发模型上结合的产物，以瀑布模型与专家系统的综合应用为基础建立的模型，该模型通过应用系统的知识和规则帮助设计者认识一个特定的软件的需求和设计，这些专家系统已成为开发过程的伙伴，并指导开发过程。

从图 1.9 中可以清楚地看到，智能模型与其他模型不同，它的维护并不在程序一级上进行，这样就把问题的复杂性大大降低了。

智能模型的主要优点有：

① 通过领域的专家系统，可使需求说明更加完整、准确和无二义性。

② 通过软件工程的专家系统，提供一个设计库支持，在开发过程中成为设计者的助手。

③ 通过软件工程知识和特定应用领域的知识和规则的应用来提供开发的帮助。

但是，要建立适合于软件设计的专家系统，或建立一个既适合软件工程又适合应用领域的知识库都是非常困难的。目前，在软件开发中正在应用 AI 技术，并已取得了局部进展；例如在 CASE 工具系统中使用专家系统，又如使用专家系统实现测试自动化。

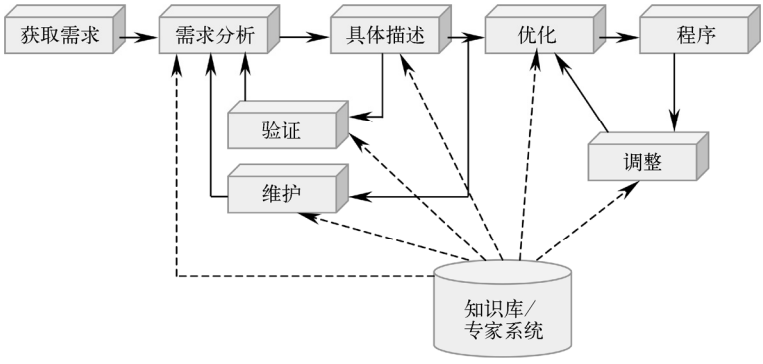

图 1.9　智能模型

1.4　软件开发方法

为了克服软件危机，从 20 世纪 60 年代末开始，各国的软件工作者一直在进行软件开发方

法的研究与实践，并取得了一系列研究成果，对软件产业的发展起着不可估量的作用。

软件工程的内容包括技术和管理两方面，且二者紧密结合。通常把在软件生命期中所使用的一整套技术的集合称为方法学（Methodology）或范型（Paradigm）。

软件开发方法是一种使用早已定义好的技术集及符号表示习惯来组织软件生产过程的方法，该方法一般表述成一系列的步骤，每一步骤都与相应的技术和符号相关。其目标是要在规定的投资和时间内，开发出符合用户需求的、高质量的软件，为此需要有成功的开发方法。

软件开发方法可分为两大类：面向过程的开发方法和面向对象的开发方法。本节将对面向过程的结构化开发方法、原型化开发方法、面向对象的开发方法及敏捷开发进行简介。

1.4.1 结构化开发方法

结构化开发方法（Structured Developing Method）是一种面向数据流的开发方法，它的基本原则是功能的分解与抽象。该方法提出了一组提高软件结构合理性的准则，如分解和抽象、模块的独立性、信息隐蔽等。它是现有的软件开发方法中最成熟、应用最广泛的方法，该方法的主要特点是快速，自然和方便。

结构化方法的指导思想是“自顶向下、逐步求精”。

结构化开发方法由三部分构成，按照推出的先后次序有：20 世纪 70 年代初推出的结构化程序设计方法——SP（Structured Program）法；20 世纪 70 年代中推出的结构化设计方法——SD（Structured Design）法；20 世纪 70 年代末推出的结构化分析方法——SA（Structured Analysis）法。SA、SD、SP 法相互衔接，形成了一整套开发方法。若将 SA 和 SD 法结合起来，又称为结构化分析与设计技术（SADT 技术）。

结构化方法的工作模型——瀑布模型（Waterfall Model），从 20 世纪 80 年代开始，逐渐发现其不足：软件开发过程是个充满回溯的过程，而瀑布模型却将其硬性分割为独立的几个阶段，不能从本质上反映软件开发过程本身的规律。此外，过分强调复审，并不能完全避免较为频繁的变动。尽管如此，瀑布模型仍然是早期开发软件产品的一个行之有效的工程模型。

1.4.2 原型化开发方法

原型反映了最终系统的部分重要特性，是一个可运行的版本，其开发基本模型如图 1.8 所示。原型化方法的基本思想是，花费少量代价建立一个可运行的系统，使用户及早获得学习的机会。原型化方法又称速成原型法（Rapid Prototyping），强调的是软件开发人员与用户的不断交互，通过原型的演进不断适应用户任务改变的需求，将维护和修改阶段的工作尽早进行，使用户验收提前，从而使软件产品更加适用。原型化方法又分为两类：

（1）快速建立需求规格原型（RSP 法）

RSP（Rapid Specification Prototyping）法所建立的原型反映了系统的主要特征，所建立的原型是需求说明书，让用户及早进行学习，不断对需求进行改进和完善，以获得更加精确的需求说明书；需求说明书一旦确定原型即被废弃，后续的工作仍按照瀑布模型开发，所以也称为废弃（Throw Away）型。

（2）快速建立渐进原型（RCP 法）

RCP（Rapid Cyclic Prototyping）法采用循环渐进的开发方式，对系统模型做连续精化，将系统需要具备的性质逐步添加上去，直至所有性质全部满足。此时的原型模型也就是最终的产品，所以也称为追加（Add On）型。

速成原型法适合于开发探索型、实验型与进化型一类的软件系统。速成原型法的工作流程

如图 1.10 所示，它是一个多次循环的过程。

在实际的软件开发过程中，通常不可能一次成功，而是一个充满反复和迭代的过程。因此，速成原型法特别适合于开发探索型、实验型与进化型一类的软件系统。而原型法的思想也符合实际的软件开发过程。

通常有三类原型：用户界面原型，功能原型，性能原型。

按照功能又可分为界面原型、功能原型和性能原型。

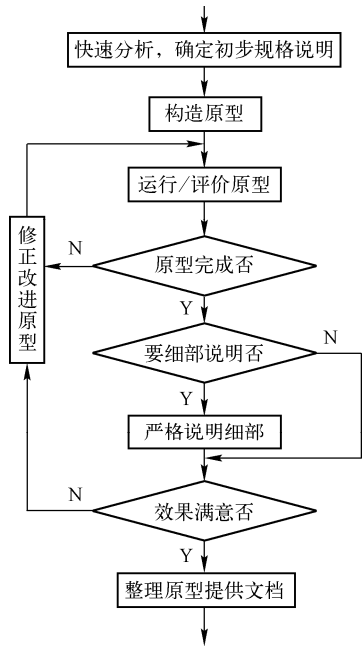

图 1.10　速成原型法工作流程

1.4.3　面向对象的开发方法

面向对象的开发（Object-Oriented Software Development，OOSD）方法是 20 世纪 80 年代推出的一种全新的软件开发方法，非常实用而强有力，被誉为 20 世纪 90 年代软件的核心技术之一。

其基本思想是：对问题领域进行自然的分割，以更接近人类通常思维的方式建立问题领域的模型，以便对客观的信息实体进行结构和行为的模拟，从而使设计的软件更直接地表现问题的求解过程。

Coad 和 Yourdon 给出一个面向对象的定义：

面向对象 = 对象 + 类 + 继承 + 消息

如果一个软件系统是按照这样四个概念来设计和实现的，则可以认为这个软件系统是面向对象的。一个面向对象的软件的每一个组成部分都是对象，计算是通过对象和对象之间的通信来执行的。

面向对象的开发方法以对象作为最基本的元素，是分析和解决问题的核心。对象与类是讨论面向对象方法的最基本、最重要的概念。

（1）对象（Object）

对象是对客观事物或概念的抽象表述，对象不仅能表示具体的实体，也能表示抽象的规则、计划或事件。通常有以下一些对象类型：

① 有形的实体：在现实世界中的实体都是对象，如飞机、车辆、机器、桌子、房子等。

② 作用：指人或组织，如教师、学生、医生、政府机关、公司、部门等所起的作用。

③ 事件：指在某个特定时间内所发生的事，如学习、演出、开会、办公、事故等。

④ 性能说明：如对产品的性能指标的说明。例如计算机主板的速度、型号、性能说明等。

每个对象都存在一定的状态（State）、内部标识（Identity）。可以给对象定义一组操作（Operation），对象通过其运算所展示的特定行为称为对象行为（Behavior）；对象本身的性质称为属性（Attribute）；对象将它自身的属性及运算“包装起来”，称为封装（Encapsulation）。因此，对象是一个封装数据属性和操作行为的实体。数据描述了对象的状态，操作可操纵私有数据，改变对象的状态。当其他对象向该对象发出消息，该对象响应时，其操作才得以实现。在对象内的操作通常叫做方法。

（2）类（Class）

类又称对象类（Object Class），是一组具有相同数据结构和相同操作的对象的集合。类是对象的模板。在一个类中，每个对象都是类的实例（Instance），它们都可以使用类中提供的函数。例如，小轿车是一个类，红旗牌小轿车、东风牌小轿车都是它的一个对象。类具有属性，

用数据结构来描述类的属性；类具有操作，它是对象行为的抽象，用操作名和实现该操作的方法（Method），即操作实现的过程来描述。

由于对象是类的实例，在进行系统分析和设计时，通常把注意力集中在类上，而不是具体的对象上。

（3）继承

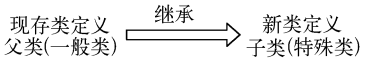

图 1.11　继承性

继承（Inheritance）以现存的定义作为基础，建立新定义的技术，是父类和子类之间共享数据结构和方法的机制。如图 1.11 所示，继承性通常表示父类与子类的关系。子类的公共属性和操作归属于父类，并为每个子类共享，子类继承了父类的特性。

继承性{单重继承：一个子类只有一个父类, 即子类只继承一个父类的数据结构和方法
多重继承：一个子类可有多个父类, 继承多个父类的数据结构和方法

通过继承关系还可以构成层次关系。单重继承构成的类之间的层次关系是一棵树，多重继承构成的类之间的关系是一个网格（如果将所有无子类的类，都看成还有一个公共子类的话）。而且继承关系是可传递的。

（4）消息

消息（message）是指对象之间在交互中所传送的通信信息。一个消息应该包含以下信息：消息名、接收消息对象的标识、服务标识、消息和方法、输入信息、回答信息等。消息使对象之间互相联系，协同工作，实现系统的各种服务。

通常一个对象向另一个对象发送信息请求某项服务，接收对象响应该消息，激发所要求的服务操作，并将操作结果返回给请求服务的对象，这种通信机制叫做消息传递。发送消息的对象不需要知道接收消息的对象如何对请求予以响应。

OOSD 由 OOA（面向对象的分析）、OOD（面向对象的设计）和 OOP（面向对象的程序设计）三部分组成。

（1）OOA（Object Oriented Analysis）法

OOA 就是要解决“做什么”的问题。它的基本任务就是要建立以下三种模型：

对象模型（信息模型）——定义构成系统的类和对象，它们的属性与操作。

状态模型（动态模型）——描述任何时刻对象的联系及其联系的改变，即时序。常用状态图，事件追踪图描述。

处理模型（函数模型）——描述系统内部数据的传送处理。

显然，在以上三种模型中，最重要的是对象模型。如何建立这三种模型，将在第 2 章中介绍。

（2）OOD（Object Oriented Design）法

在需求分析的基础上，进一步解决“如何做”的问题。OOD 法也分为概要设计和详细设计。

其中面向对象的分析（OOA）与面向对象的设计（OOD）是面向对象开发方法的关键。

由于面向对象的方法以对象为核心，强调模拟现实世界中的概念而不是算法，尽量用符合人类认识世界的思维方式来渐进地分析、解决问题，对软件开发过程所有阶段进行综合考虑，能有效地降低软件开发的复杂度，使软件的易复用性和易扩充性都得到了提高，而且能更好地适应需求的变化，提高软件质量。

1.4.4　敏捷软件的开发

1．敏捷软件开发的基本概念

敏捷软件开发又称敏捷开发，是以用户的需求进化为核心，采用迭代、循序渐进的方法进

行软件开发。20 世纪 90 年代，软件危机得到一定程度的缓解，但随着软件项目规模和复杂度的增加，需求常常发生变化，时有软件不能如期交付的情况发生。为了按时交付软件，开发人员只好经常加班加点赶进度，而大量的文档资料也加重了他们的负担。

激烈的市场竞争也要求推出快速、高质量开发软件的方法，因此敏捷软件开发方法便应运而生。2001 年 2 月部分软件工作者在美国犹他州成立了“敏捷软件开发联盟”（Agile software development），简称 Agile 联盟，发表了敏捷软件开发宣言，表述了与会者对软件开发的核心价值观：

（1）人和交互　　胜过　过程和工具

（2）可运行的软件　　胜过　面面俱到的文档

（3）与客户协作　　胜过　合同谈判

（4）对变更及时处理　胜过　遵循计划

可以看出，敏捷开发更强调与客户的协作、人与人之间的交互与团队的协作，更重视不断向用户提交可运行的软件，而不把过多的精力放在编写详尽的文档上。尤其强调对软件需求变化的快速应变能力。

在这些价值观的指导下，提出了敏捷软件开发必须遵守的 12 条原则：

（1）最重要的是要尽早和不断提交有价值的软件以满足客户需求。

（2）欢迎需求的变化，即使是在开发的后期，敏捷过程也能利用变化来为客户提升竞争优势。

（3）几周或几个月，经常提交可运行的软件，时间间隔越短越好。

（4）在整个项目过程中，业务人员和开发人员必须每天在一起工作。

（5）围绕有工作激情的人建立的项目组，给予他们所需的环境和支持，并对他们能够完成任务予以充分信任。

（6）项目组内最有效、效率最高的信息传递方式是面对面的交流。

（7）可运行的软件是度量项目进度的首要标准。

（8）敏捷过程提倡可持续开发，项目责任人、开发者和用户应保持长期稳定的开发速度。

（9）不断追求优秀的技术和优良的设计，有助于提高敏捷性。

（10）简单化是有效降低工作量的艺术。

（11）最好的架构、需求和设计源于自我组织的团队。

（12）团队要定期进行反省，讨论如何能够更有效地工作，并对工作进行相应调整。

2．XP 方法简介

按照敏捷软件开发的思想和原则，推出了许多具体的实践方法，如：XP、Scrum、Crystal、Methods、FDD 等。

其中 XP 方法是最具代表性的敏捷开发方法，又称极限编程（Extreme Programming，XP）。它是由 kent Beck 于 1999 年提出来的。极限编程以用户需求作为软件开发的最终目标，是一种以实践为基础的软件工程过程。极限编程强调测试，是一种测试驱动的开发方法，强调代码质量和及早发现问题，以适应环境和需求的变化。

（1）核心价值观

XP 方法的核心价值观为：沟通（Communication）、简单（Simplicity）、反馈（Feedback）和勇气（Courage）。

沟通——是项目成功的关键，只有开发人员与用户、开发人员之间频繁而有效的面对面信息交流，充分理解用户需求，就能够保证软件开发的质量和效率。

简单——为了保证高效的开发，在满足用户需求的前提下，软件开发全过程及过程中的产品都应该尽量简单。

反馈——及时、准确的信息反馈，能够及时发现开发工作中的问题和偏差并及时纠正。

勇气——采用敏捷开发这种新的开发方法，就是一种挑战，是需要勇气的；在开发过程中需要团队密切协作，既要相信别人也要相信自己一定能够完成，这需要勇气。另外，如只有十分需要的文档才写，即使写也要简单明了，这也需要勇气。

（2）XP 方法的最佳实践

在其核心价值观的指导下，XP 方法提出 12 项最佳实践：

① 规划策略（The Planning Game）。通过结合使用业务优先级和技术评估来快速制定计划，确定下一个版本的范围。

② 小型发布（Small Release）。将一个简单系统迅速投入生产，以很短的周期发布新版本，供用户评估使用。

③ 系统隐喻（System Metaphor）。用合适的比喻传达信息，通过隐喻来描述系统如何运作、新的功能以何种方式加入系统，通常包含了一些可以参照和比较的类和设计模式。

④ 简单设计（Simple Design）。任何时候都应当将系统设计为尽可能简单。不必要的复杂性一旦被发现就马上去掉。

⑤ 测试（Testing）。程序员不断地进行单元测试，在这些测试能够准确无误地运行的情况下，开发才可以继续。客户编写测试来证明系统各功能都已经完成。

⑥ 重构（Refactoring）。程序员重新构造系统以去除重复、改善沟通、简化或提高系统柔性。

⑦ 结对编程（Pair programming）。所有的生产代码都是由两个程序员在同一台机器上编写的，这样能够随时交流，及时发现和解决问题。

⑧ 代码集体所有（Collective code ownership）。任何人在任何时候都可以在系统中的任何位置更改任何代码。

⑨ 持续集成（Continuous Integration）。每天多次集成和生成系统，每次都完成一项任务。

⑩ 每周工作 40 小时（40-hour Week）。一般情况下，一周工作不超过 40 小时。不要连续两个星期都加班。

⑪ 现场客户（On-site Customer）。在团队中加入一位真正的、起作用的用户，他将全职负责回答问题。

⑫ 编码标准（Code Standards）。程序员依照强调通过代码沟通的规则来编写所有代码。

这 12 项最佳实践，就是强调开发者之间、开发者与用户之间的充分交流、密切协作，快速、高效地不断测试、集成和推出系统。

（3）XP 方法的开发过程

XP 使用面向对象方法作为推荐的开发范型。XP 包含了策划、设计、编码和测试 4 个框架活动的规则和实践。图 1.12 描述了 XP 开发过程，并指出与各框架活动相关的关键概念和任务。特别要说明，XP 开发过程的主要特点是一个不断迭代的过程。

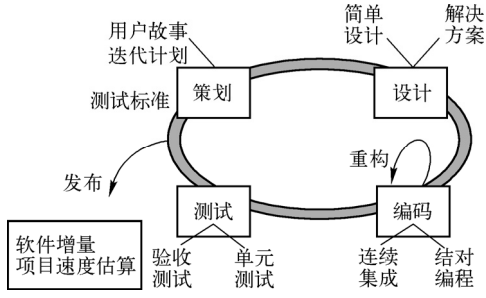

图 1.12　XP 编程过程

（4）敏捷开发的原则

① 快速迭代。在敏捷开发中，软件项目在构建初期被分成多个子项目，各个子项目的成果都经过测试，具备可视、可集成和可运行使用的特征。也就是把一个大项目分为多个相互联

系，但也可独立运行的小项目，并分别完成，在此过程中软件一直处于可使用状态。

相对于那种半年一次的大版本发布来说，小版本的需求、开发和测试更加简单快速。

② 让测试人员和开发者参与需求讨论。需求讨论以小组的形式展开最有效率。且必须要包括测试人员和开发者，这样可以更加轻松定义可测试的需求，将需求分组并确定优先级。同时，该种方式也可以充分利用团队成员间的互补特性。如此确定的需求往往比开需求讨论大会的形式效率更高，大家更活跃，参与感更强。

③ 编写可测试的需求文档。开始就要用“用户故事”（User Story）的方法来编写需求文档。这种方法可以让我们将注意力放在需求上，而不是解决方法和实施技术上。过早地提及技术实施方案，会降低对需求的注意力。

④ 多沟通，尽量减少文档。任何项目中，沟通都是一个常见的问题。好的沟通，是敏捷开发的先决条件。在圈子里面混得越久，越会强调良好高效的沟通的重要性。

团队要确保日常的交流，面对面沟通比邮件强得多。

⑤ 做好产品原型。建议使用草图和模型来阐明用户界面。并不是所有人都可以理解一份复杂的文档，但人人都会看图。

⑥ 及早考虑测试。及早地考虑测试在敏捷开发中很重要。传统的软件开发，测试用例很晚才开始写，这导致过晚发现需求中存在的问题，使得改进成本过高。较早地开始编写测试用例，当需求完成时，可以接受的测试用例也基本一块完成了。

1.5 软件工具与集成化开发环境

软件工具是用于辅助软件开发、运行、维护、管理等活动的软件系统，使用功能强大、方便适用的软件开发工具可以降低软件开发和维护的成本，提高软件生产效率，改善软件产品的质量，所以软件工具是软件工程研究的重要内容之一。

在软件开发过程中，软件工程师和管理人员按照软件工程的方法和原则，借助于软件工具进行开发、维护、管理软件产品的过程，称为计算机辅助软件工程（Computer-Aided Software Engineering，简称 CASE）。CASE 的实质是为软件开发提供一组优化集成的且节省大量人力的软件开发工具，其目的是实现软件生存周期各环节的自动化并使之成为一个整体。

1.5.1 软件工具的发展过程

CASE 发展经历了两个阶段：

1．依赖于软件生命周期各阶段的分散工具

在软件工程的早期应用的是孤立的单个软件开发工具，支持软件开发过程中的某一项特定活动，这类工具通常有不同的用户界面和数据存储格式，它们之间彼此独立，不能或很难进行通信和数据的共享与交换，不能有效支持软件开发的全部过程。

2．软件开发环境

另一类软件工具是集成化的 CASE 环境，是在克服孤立软件工具缺陷的过程中发展起来的，它将在软件开发过程不同阶段所使用的工具进行集成，使其具有一致的用户界面和可共享的信息数据库。

CASE 工具在发展过程中逐渐形成了能够支持软件生存周期各阶段的工具，称为软件开发环境（Software Development Environment，SDE），也称为软件工程环境（Software Engineering

Environment），是包括方法、工具和管理等多种技术在内的综合系统。是为支持系统软件和应用软件的工程化开发和维护而使用的一组软件。良好的软件开发环境能够简化软件开发过程，提高软件开发质量。SDE 应具备以下特点：

① 紧密性：各种工具紧密配合工作；

② 坚定性：环境可自我保护，不受用户和系统影响，可实现非预见性的环境恢复；

③ 可适应性：适应用户要求，环境中的工具可修改、增加、减少；

④ 可移植性：指工具可移植。

如图 1.13 所示，典型的软件工程环境可具有三级结构：核心级，包括核心工具组、数据库、通信工具、运行支持、功能、与硬件无关的移植接口等；基本级，包括环境的用户工具，编译、编辑程序，作业控制语言的解释程序等；应用级，通常指应用软件的开发工具。

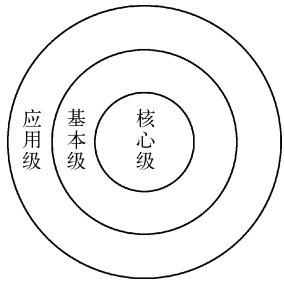

图 1.13　典型的软件工程环境

1.5.2　软件工具

软件工具可按不同方式进行分类，如按软件开发模型及开发方法分类，按功能及结构特点分类，按开发阶段分类等。下面按照软件工程过程，介绍用于软件开发的工具，软件维护的工具和软件管理与支持的工具。

1．软件开发工具

软件开发工具按照软件开发阶段分为以下几种。

（1）分析工具

分析工具用于辅助软件开发人员完成软件系统需求分析的活动，包括根据需求的定义，生成准确而完整、清晰、一致的功能规范。软件分析工具包括三种类型：基于自然语言或图形描述的需求分析工具；基于形式化需求定义语言的工具和其他需求分析工具。

典型的有 Rational 公司的 Analyst Studio 成套的需求分析工具软件，是用于应用问题分析和系统定义的一组相对完备的工具集，适合于团队联合开发使用。

（2）设计工具

设计工具是用于帮助软件开发人员完成软件系统的设计活动的软件系统，根据需求阶段获得的功能规范，生成与之对应的软件设计规范。通常，软件设计工具包括基于图形描述、语言描述的设计工具；基于形式化描述的设计工具；面向对象的设计工具三种类型。

典型的有 Enterprise Architect，是一个基于 UML 的 Visual CASE 工具，主要用于设计、编写、构建和管理以目标为导向的软件系统。

（3）编码工具

编码工具主要包括：编辑程序、汇编程序、编译程序和调试程序等。这些编码工具可以是彼此独立的应用程序，也可以是一个集成的程序开发环境，集成了源代码的编辑程序、编译程序和链接程序，以及用于源代码排错的调试程序和可供发布产品的发布程序。

典型的集成程序开发环境有 Microsoft 公司的 Visual C++、Visual Basic 和 Borland 公司的 Delphi、C++ Builder 等。

（4）调试工具

调试工具也称排错工具，用于及时发现和排除程序代码中的错误和缺陷，调试工具又分为源代码调试程序和调试程序生成程序两类。

源代码调试程序用于了解程序的执行状态和查询相关数据信息，发现和排除程序代码中存

在的错误和缺陷。一般由执行控制程序、执行状态查询程序和跟踪包组成。

执行控制程序用于断点定义、断点撤销、单步执行、断点执行、条件执行等功能。执行状态查询程序用于了解程序执行过程中 CPU、寄存器、堆栈、变量等数据结构中存储的数据与信息。跟踪包则用于跟踪程序执行过程中所经历的事件序列。

调试程序生成程序是一种通用的调试工具，能针对给定的程序设计语言，生成相应的源代码调试程序。

2．软件维护工具

软件维护的主要任务是在软件产品投入运行以后，纠正软件开发过程中未发现的错误，改进和完善软件的功能和性能，以适应用户新的需求，延长软件产品的使用寿命。主要的软件维护工具包括：

（1）版本控制工具

版本控制工具对在软件开发过程中所产生的不同版本进行存储、更新、恢复和管理。

典型代表有 UNIX 操作系统的 SCCS（源代码控制系统）。SCCS 为一个源代码文件的所有版本建立一棵版本树，每一个版本都是该版本树的一个节点。SCCS 完整存储该文件的第一个版本，而后续的其他版本则只存储它与以前版本的不同之处。SCCS 通过版本树维护管理各个版本的更新历史，可恢复到以前的任何一个版本。

（2）文档管理工具

软件开发过程中所产生的大量文档，其编写通常花费开发工作量的 20%到 30%。文档管理工具用于对软件文档进行分析、组织、维护和管理，这对提高软件开发的质量和效率具有重要意义。

（3）开发信息库工具

开发信息库工具用于记录保存项目开发的相关信息，如每个对象的开发与修改信息；维护对象和与之相关信息间的关系，记录对象的开发人员、新版本对象中发生的改动、对象中存在的错误、对该对象进行测试时使用的测试用例、测试结果之间的关系等；还记录用来生成此软件产品的所有开发工具的版本信息、所采用的程序设计语言和应用程序开发接口。

（4）逆向工程工具

软件的逆向工程是指对已有的软件进行分析，获取比源代码更高级的表现形式，如提取出数据结构、体系结构、程序总体设计等各种有用的软件开发信息。早期的逆向工程工具有反汇编工具、反编译工具等。现在的逆向工程工具能够分析高级程序设计语言的源程序，恢复程序的控制结构、流程图、PAD 图等更高级的抽象信息，为软件的理解和维护提供方便。

（5）再工程工具

所谓再工程是指在通过逆向工程获得软件设计等信息的基础上，利用这些信息修改或重构软件系统，增加新的功能和改进性能。

再工程工具用来辅助软件开发人员重构一个功能和性能更为完善的软件系统。目前，再工程工具的使用主要集中在代码重构、程序结构重构和数据结构重构等方面。

3．软件管理与支持工具

软件产品的管理与支持是软件能否开发成功的关键。软件管理与支持工具用于确保软件产品的质量和软件产品的开发效率。这类工具主要包括：

（1）软件评价工具

软件评价工具对于实现软件产品的质量控制，确保软件产品的正确性、可靠性具有十分重要的意义。软件评价工具根据某个软件质量模型，例如 ISO 软件质量度量模型、McCall 软件

度量模型，对软件产品的质量、复杂性加以度量，并形成该软件产品的质量评价报告。

（2）软件配置管理工具

在软件产品的开发过程中，变动和修改是不可避免的。在对软件产品进行修改前，必须进行相应的分析论证，确保修改的质量和正确性，并在修改后加以记录。软件配置管理工具可对软件配置项进行标示、版本控制、审计和状态统计等，使对各配置项的访问、修改易于实现，简化审计过程、改进状态统计、减少软件错误、提高软件质量。

（3）软件项目管理工具

软件项目管理工具是对软件产品的开发活动进行有效的管理。包括软件项目所涉及的人员、费用、进度和质量四个方面的有效管理。如其中重要的成本估算工具，是根据某估算模型，例如 Halstead 模型、Putnam 模型、COCOMO 模型等，对软件项目的成本进行估算。

（4）风险分析工具

风险管理对于一个大型项目是极为重要的。风险分析工具可以通过提供对风险标示和分析的详细指南，使得项目管理者能够有效地对在软件项目开发过程出现的风险进行控制和规避。

1.5.3 集成化 CASE 环境

集成化 CASE 环境是将多个 CASE 工具结合起来，使得各种软件开发信息能够在不同 CASE 工具之间、不同开发阶段之间，以及不同开发人员之间顺畅传递。按照集成度的高低可分为以下几种层次：

（1）具有信息传递的软件工具集

工具间是完全独立的，它们之间有着不同的用户界面和信息的存储格式。如图 1.14 所示，它们借助于操作系统的文件服务和数据交换服务使得工具 A 的输出文件能够被导入到工具 B 中，借助于此种方式实现不同工具之间的数据交换和共享，集成度较低。

随着开发工具的数量增加，每种开发工具使用不同的文件格式进行信息的存储，文件格式之间转换将变得非常复杂，并且反复进行格式转换可能导致文件信息的一致性和完整性遭到破坏。

（2）具有公共界面的软件工具集

如图 1.15 所示，这些软件工具集为使用者提供了一致的公共用户界面和操作方式，如相同的菜单、工具按钮、快捷方式等，为软件开发人员提供了极大的便利。但软件工具之间的数据交换仍然沿用了在不同格式的文件导入/导出的方式，严重影响了它们之间数据交换的效率和数据的完全性与完整性。

图 1.14　具有信息传递的软件工具

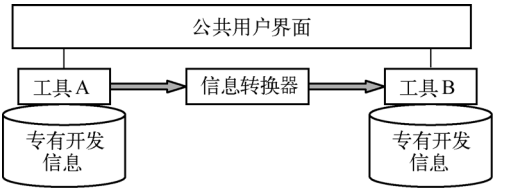

图 1.15　具有公共界面的软件工具集

（3）信息共享的软件工具集

工具集之间不仅具有一致的用户界面和操作方式，而且对不同工具的开发信息进行统一的存储和管理。如图 1.16 所示，这种信息共享的集成方式从根本上解决了在不同的软件工具之间进行信息交换的问题，提高了工具之间的继承度。同时在不同的软件工具之间具有共同的信息存储的标准。

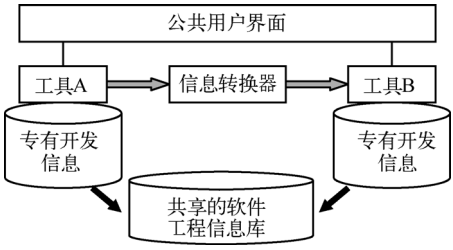

图 1.16　信息共享的软件工具集

小　　结

本章为读者阅读本书进行必要的知识准备；介绍了软件工程的基本概念；软件危机与软件工程的产生与发展，软件生存周期，软件过程。对典型的软件过程模型进行了介绍，包括瀑布模型、循环模型、螺旋模型、喷泉模型及智能模型。本书的重点是讨论面向对象的开发方法及采用 UML 统一建模语言建立系统模型，本章仅结合软件过程模型对结构化开发方法、原型化开发方法和面向对象的开发方法，以及敏捷开发方法的基本概念和 XP 极限编程进行了简介。

最后，对于软件工具及集成化开发环境进行了介绍。尤其是 CASE 集成环境是集成化程度最高的软件开发工具，它把各种软件工具有机地集成在一起，做到了界面集成、数据集成、控制集成。其特征是支持软件开发的各个阶段，强调各种工具在各开发阶段中的特殊作用，各种软件开发工具涵盖了整个开发过程。

习　题　1

一、选择题

1．软件的主要特性是（　　）。

（A）无形　　（B）高成本

（C）包括程序和文档　　（D）可独立构成计算机系统

2．软件工程三要素是（　　）。

（A）技术、方法和工具　　（B）方法、工具和过程

（C）方法、对象和类　　（D）过程、模型、方法

3．软件危机的主要表现是（　　）。

（A）软件成本太高　　（B）软件产品的质量低劣

（C）软件开发人员明显不足　　（D）软件生产率低下

4．软件工程的主要目标是（　　）。

（A）软件需求　　（B）软件设计

（C）风险分析　　（D）软件实现

5．包含风险分析的软件工程模型是（　　）。

（A）螺旋模型　　（B）瀑布模型

（C）增量模型　　（D）喷泉模型

6．下面属于面向对象开发方法的有（　　）。

（A）Booch　　（B）UML　　（C）Coad　　（D）OMT

7．软件开发方法的主要工作模型有（　　）。

（A）螺旋模型　　（B）循环模型　　（C）瀑布模型　　（D）专家模型

8．软件工程的目标有（　　）。

（A）易于维护　　（B）低的开发成本　　（C）高性能　　（D）短的开发期

9．软件工程的目的和意义是（　　）。

（A）应用科学的方法和工程化的规范管理来指导软件开发

（B）克服软件危机

（C）做好软件开发的培训工作

（D）以较低的成本开发出高质量的软件

二、判断题

1．软件就是程序，编写软件就是编写程序。（　　）

2．瀑布模型的最大优点是将软件开发的各个阶段划分得十分清晰。（　　）

3．结构化方法的工作模型是使用螺旋模型进行开发。（　　）

4．结构化方法和 OO 方法都是一种面向过程的软件开发方法。（　　）

5．原型化开发方法包括生成原型和实现原型两个步骤。（　　）

6．面向对象的开发方法包括面向对象的分析、面向对象的设计和面向对象的程序设计。（　　）

7．软件危机的主要表现是软件的需求量迅速增加，软件价格上升。（　　）

8．软件工具的作用是延长软件产品的寿命。（　　）

9．软件工程过程应该以软件设计为中心，关键是编写程序。（　　）

10．RCP 法与 RSP 法的主要区别是，前者采用循环渐进的开发方式，原型将成为最终的产品，而后者将被废弃。（　　）

三、简答题

1．软件产品的特性是什么？

2．软件发展有几个阶段？各有何特征？

3．什么是软件危机？其产生的原因是什么？

4．什么是软件过程？有哪些主要的软件过程模型？它们各有哪些特点？

5．有哪些主要的软件开发方法？

6．软件生命期各阶段的主要任务是什么？

7．原型化方法的核心是什么？它具有哪些特点？

8．面向对象的开发方法为什么逐渐成为软件开发的主流方法？

9．什么是软件开发环境？它对软件开发过程有何意义？

10．敏捷软件开发的核心思想是什么？以 XP 方法为例进行说明。

11．软件开发工具的集成可以分成哪几个层次？

12．集成化的 CASE 环境相对于彼此独立的软件开发工具有哪些明显的优势？