第 1 章　软件危机与软件工程

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。

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，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。

20 世纪 60 年代以来，随着计算机的广泛应用，软件生产率、软件质量远远满足不了社会发展的需求，成为社会、经济发展的制约因素。当时软件开发虽然有一些工具支持，例如编译连接器等，但基本上还是依赖开发人员的个人技能，没有可遵循的原理、原则和方法，也缺乏有效的管理。软件可靠性、可维护性较差，而且往往超出预期的开发时间要求。

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”。

这一章介绍软件危机和软件工程的基本概念。

1.1　软件危机

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。

20 世纪 60 至 70 年代，“软件危机”一词在计算机界广为流传，其主要针对当时存在的软件代价高和软件错误多的现象。

1.1.1　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礼之后，由于离预定目标甚远，宣告失败。

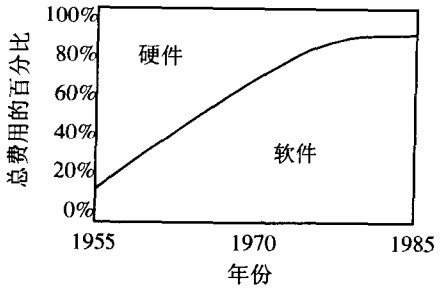

图 1.1　计算机系统硬件／软件成本变化趋势

1.1.2　软件开发和维护中严重问题

在软件开发过程中的严重问题：在工业、商业、科学技术和国防等部门需要的软件功能很强，往往有几万、几十万、甚至几百万行代码。完成这样一个系统，在一定的时间内一个人或几个人的智力与体力是承受不了的。由于软件是逻辑、智力产品，盲目增加软件开发人员并不能成比例地提高软件开发能力。相反，随着人员数量的增加，人员的组织、协调、通信、培训和管理等方面的问题将更加严重。

在软件运行中的问题：系统运行过程中所暴露出来的软件错误会造成两个方面的损失：

（1）使系统受到了损害。

（2）随之而来的纠正工作。

例如在一个银行的电子存款——转帐系统中，假定有位客户在开列新帐户的当天又撤消了它。如果系统只是成功地开列了帐户而没有及时将其撤消，那么很明显，就会产生多付款的现象。当一位银行管理人员遇到这种错误情况时，可以有下列四种不同的选择：

（1）这种事情难得发生，容易鉴别，因此错误被遗忘。

（2）由于纠正错误要付出较高的代价，发生的可能性又小，因此不再修改软件，只是通知全体有关人员注意这种异常现象的发生。

（3）在适当的时候对软件的错误予以纠正。

（4）立即指定一名软件设计人员进行诊断，重新设计，测试并纠正错误。

显然，方法 2 或方法 3 对这种场合比较合适（也适用于大部分场合），方法 1 和方法 4 就很不合适，而且是不经济的。为了排除错误就要花费一定的代价，修复错误所需的相对成本与软件开发与使用的阶段有关，随着项目的进展，软件修改日趋复杂，排错代价越来越高。

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开发和维护有关的问题远不止这些。

1.1.3 产生软件危机的原因

产生软件危机的原因很多，除了与软件本身固有的特征有关以外，还与软件开发范型、软件设计方法、软件开发支持以及软件开发管理等有关。

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。

大型软件项目开发需要组织一定的人力共同完成，各类人员的信息交流不及时、不准确、甚至产生误解，这也是产生软件危机的主要原因。

此外，如果软件运行中发现的错误，很可能是在开发时期引入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，轻视软件维护，也是造成软件危机的一个重要原因。

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。

针对以上原因，建立计算机软件开发和维护的正确概念是解决软件危机的正确途径。

1.1.4 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。

（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，就是为了解决软件开发过程中的技术和管理问题。

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 软件工程的定义与基本原理

软件工程学科是一门指导计算机软件开发和维护的工程学科。软件工程是一类求解软件的工程。它应用计算机科学、数学及管理科学等原理，借鉴传统工程的原则、方法来生产软件以达到提高质量，降低成本的目的。其中，计算机科学、数学用于构造模型与算法，工程科学用于制定规范设计范型、评估成本及确定权衡，管理科学用于计划、资源、质量、成本等控制。软件工程的方法、工具、过程构成了软件工程的三要素。

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，提出了软件工程的七条基本原理。这七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 认为在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，使通讯开销减少。

（7）承认不断改进软件工程实践的必要性

不仅要积极主动地采纳新的软件技术，而且要不断总结经验，以供今后的软件开发借鉴。

1.2.2 软件工程的目标

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念列如表 1.1所示。

表 1.1 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成功运行的比率
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）	软部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（tracebility）	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。尤其对于实时嵌入式计算机系统，可靠性是一个非常重要的目标。因为软件要实时地控制一个物理过程，如宇宙飞船的导航、核电站的运行等等。如果可靠性得不到保证，一旦出现问题可能是灾难性的，后果不堪设想。因此，在软件开发、编码和测试过程中，必须将可靠性放在重要地位。

12.3 软件工程框架及原则

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四条基本原则，如图 1.2所示。

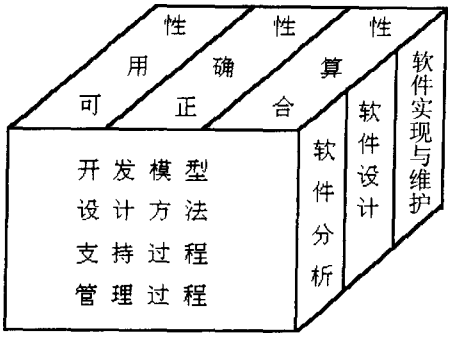

图 1.2 软件工程的框架

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、硬件需求以及其它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 本章小结

本章介绍软件危机与软件工程的基本概念。软件工程的目标指明项目追求的和应该达到的标准。软件工程的框架概括了如何从开发范性、设计方法、支持过程、管理过程等方面入手，保证软件工程的正确性、可用性、经济性。本章内容指导软件工程的项目研究与管理，是软件工程基本概念。

习　　题

1.1 试论述“轮件危机”产生的原因和解决方法。

1.2 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？

1.3 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。

1.4 软件工程学的基本原则有哪些？为什么？

第 2 章　软件生命周期及软件开发模型

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。软件开发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。

本章内容：首先阐述软件生命周期，然后介绍软件开发模型。

2.1　软件生命周期

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阶段。目前各阶段的划分尚不统一，但无论采用哪种划分方式，都包括软件定义、软件设计、软件使用与维护三阶段，而又可以具体分成几个子阶段。

1. 软件定义

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。

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，编写出软件需求说明书及系统用户手册，交管理机构评审。

2. 软件分析

总体设计：又称概要设计，是软件工程的技术核心。在此阶段，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。

详细设计：对每个模块要完成的工作进行具体的描述，为源程序的编写打下基础。

3. 软件使用与维护

编码（实现）：把软件设计转换成计算机可以接受的程序代码，软件使用阶段写出的程序应当是结构良好并与设计相一致的。

软件测试、运行／维护：对模块在功能和结构问题加以纠正，并进行组装测试与确认测试。软件投入使用后，还要进行维护。所谓软件的退役是终止对软件的支持，即停用。

研究软件生命周期是为了更科学、有效地组织和管理软件的生产，从而使软件生产更可靠、更经济。

2.2　软件模型

软件开发模型是软件开发全部过程、活动和任务的结构框架。图 *2.1* 软件生命周期软件

开发模型能清晰、直观地表达软件开发全部过程，明确规定要完成的主要活动和任务，用来作为软件项目工作的基础。模型应该是稳定和普遍适用的。

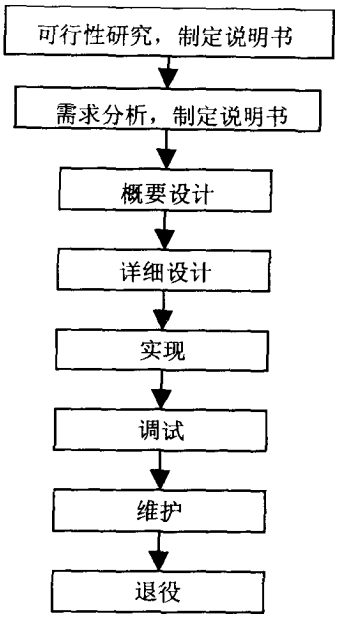

图 2.1　软件生命周期

最早出现的软件开发模型是 1970 年 W. Royce 提出的瀑布模型，而后随着软件工程学科的发展和软件开发的实践，相继提出了原型模型、演化模型、增量模型、喷泉模型等。

2.2.1　瀑布模型

瀑布模型将软件生存周期的各项活动规定为依序连接的若干阶段工作，形如瀑布流水，最终得到软件产品。

瀑布模型可追溯到 20 世纪 50 年代末期，当时人们已感到必须先确认“做什么”，才能编制程序将其实现，即使是比较简单的小型问题也不例外。对于较大的软件项目，问题更加复杂，一个精确的软件开发步骤按需要解决问题的顺序依次为：做什么——如何做——制作检测——使用，于是一个反映软件过程的基本框架如图 2.2所示。

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。如评审过程发现错误和疏漏，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（见图 2.3）。模型中各个阶段的任务和软件开发活动如上节所述。

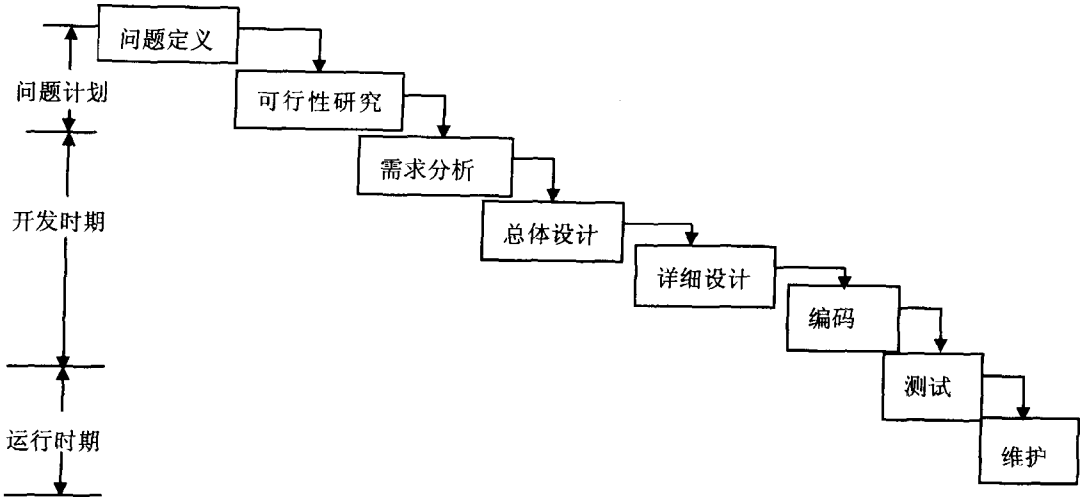

图 2.2 瀑布模型

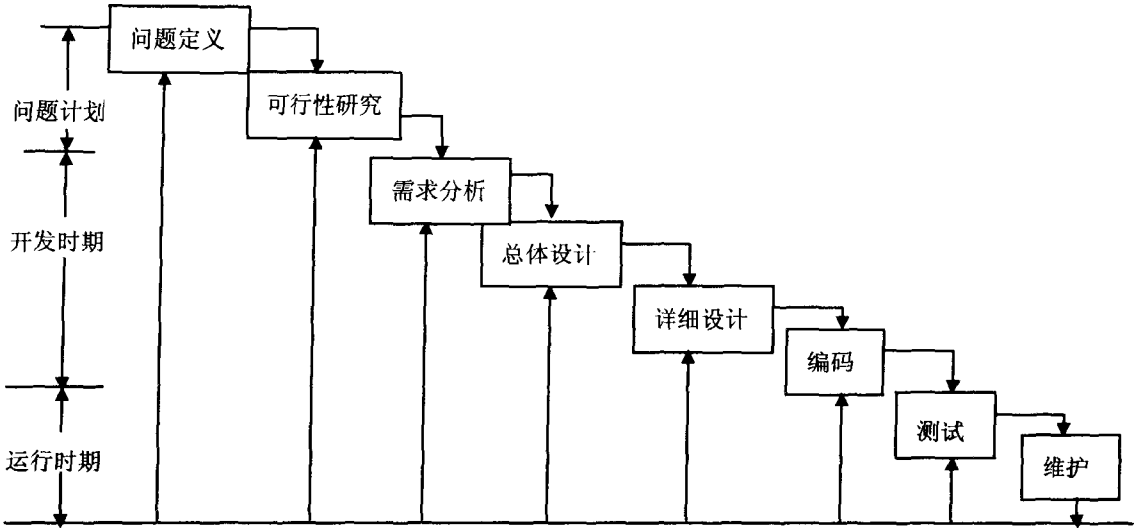

图 2.3 带有反馈的瀑布模型

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起着显著作用。它提供了软件开发的基本框架，这比依靠“个人技艺”开发软件好得多。它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，从而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，为此必须付出高额的代价，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。因此，瀑布模型的应用有一定的局限性。

2.2.2 演化模型

演化模型主要针对用户事先不能完整定义需求的软件开发项目，在逐个项目实施工程中需要用户与软件开发人员的密切配合。用户可以先给出待开发系统的核心需求，当看到核心需求实现后，能够有效地提出反馈，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，并提出精化系统、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。如图 2.4 所示。

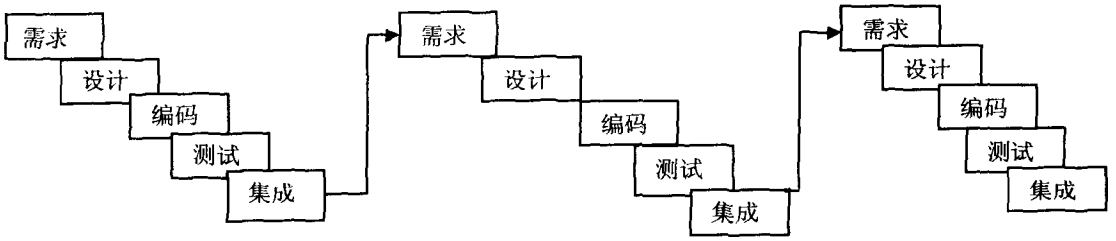

图 2.4 演化模型

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。演化在一定程度上减少了软件开发活动的盲目性，但整个软件开发活动应持续进行，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。

2.2.3 原型模型

原型（prototype）模型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中，进一步修改、完善、确认软件系统的需求并达到一致的理解。

快速开发原型的过程是：

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（或界面形式、或处理功能、或总体结构、或模拟性能等），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，如数据的输入方式、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。

原型开发模型如图 2.5 所示。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。

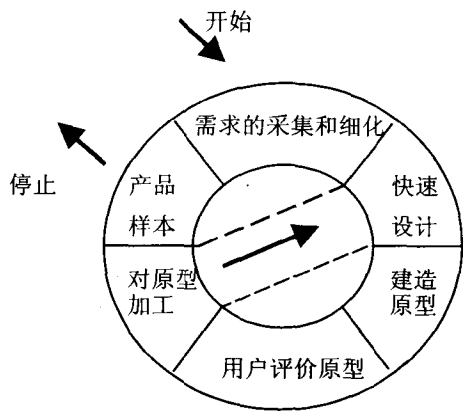

图 2.5　原型开发模型

2.2.4　螺旋模型

螺旋模型(spiral model）是生存周期模型与原型模型的结合，不仅体现了两个模型的优点，而且还增加了新的成分——风险分析。螺旋模型的结构如图 2.6 所示，它由四个部分组成：需求定义；风险分析；工程实现；评审。

螺旋模型是由上面四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、工程实现和评审四个阶段。

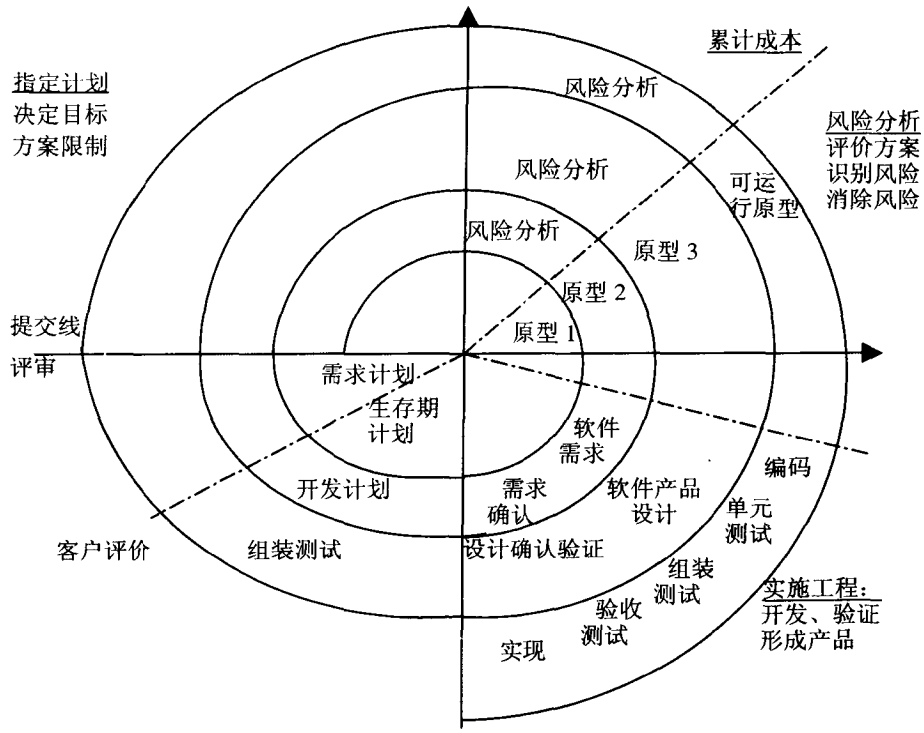

图 2.6　螺旋模型的结构

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一个系统而不是一个初始版本。

2.2.5　喷泉模型

“喷泉”一词本身体现了迭代和无间隙特性。喷泉模型如图 2.7 所示。

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。

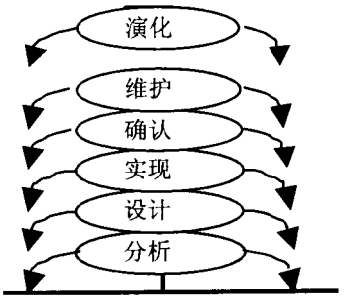

图 2.7　喷泉模型

同时，该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。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。

2.3　本章小结

软件生命周期实质上是大型系统开发过程中各项目阶段的一种表示方法，如同任何事物一样，软件也有一个孕育、诞生、成长、成熟、衰亡的生存过程。根据这一思想，把上述基本的过程活动进一步展开，可以得到软件生命期的 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。

习　　题

1. 说明“软件生命周期”的概念。
2. 试论述瀑布模型软件开发方法的基本过程。
3. 举例说明哪些项目的开发适用于原型与螺旋模型，哪些不适用于采用这两种模型。

第 3 章　计算机系统工程

计算机软件工程和硬件工程可以看作是一门更广义的学科——“计算机系统工程”内的活动。它们所要做的都是按一定的次序开发基于计算机的系统。

计算机系统工程是指与构造基于计算机系统有关的过程、方法和技术。它是一种问题求解活动。计算机系统工程的任务是：组织并指导系统工程师定义全系统各层次中的所有基于计算机系统的要素。

本章主要内容：首先阐述基于计算机的系统，包括：计算机系统工程；硬件和硬件工程；软件和软件工程；人机工程；数据库和数据库工程。

然后介绍可行性研究：可行性研究的任务；经济可行性；技术可行性；方案选择。

最后阐述系统结构的模型化。

3.1　基于计算机的系统

基于计算机的系统是“某些要素的一个集合，这些要素被组织起来以实现某种方法、过程或借助处理信息进行控制。”图 3.1 给出了基于计算机系统的系统要素及相互之间关系。

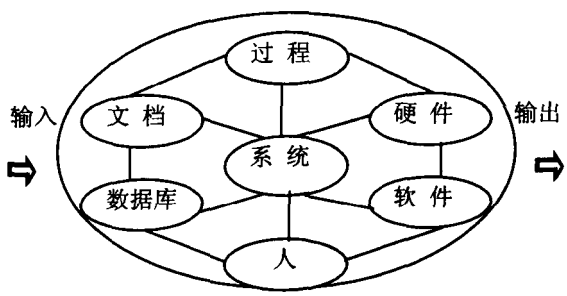

图 3.1　基于计算机系统的系统要素及相互之间关系

在计算机系统中：

“软件”是指计算机程序、数据结构、用以描述所需要的逻辑方法、过程或控制的相关文档。软件必须在计算机系统的支持下才能工作。

“硬件”是指所有提供计算能力的电子设备以及进行数据采集与处理的设备。

“人”是指硬件和软件的用户和操作员，无论系统的自动化程度多么高，都需要人的参与。

“数据库”是一个大型的信息的有组织的集合，收集并组织与系统有关的信息。它通过软件进行存取，是系统功能的一个主要部分。

“文档”是指使用手册及必要的表格和其他用以描述系统使用和操作的描述性信息，用于指导系统的使用。

“过程”则是定义每一种系统要素的特定使用的步骤，或系统驻留的过程性环境。

以上这些系统要素可以按各种方式进行组合以进行信息转换，例如，机器把包含一些特定指令的命令文件转换成一组控制信号，产生某些特定的物理动作。

如果不考虑系统内部结构及功能，基于计算机的系统要素可用 IPO 模型表示，其中 I 指信息的输入，P 指信息的处理，O 指信息的输出。

多数大型的基于计算机的系统其要素本身也是一个基于计算机的系统。因此，大型计算机系统具有复杂的层次结构。

3.1.1 计算机系统工程

计算机系统工程是一个问题求解活动，目的是揭示、分析所期望的功能，并把它们分配到各个单独的系统要素中去。

在系统的功能规范确立之后，下一步要对系统的功能进行认真、仔细的分析，问题的焦点集中于功能、性能、信息流和容量上。

系统分析员不是向用户询问这个任务该如何去做，而是应当询问用户需要什么。然后，系统分析员应设想合理的回答，并提出一些候选的分配方案。在每一个候选方案中，系统的功能和性能都被分配给各种不同的系统生成要素中，以制定实现这些功能的系统总体方案。

如果说系统的功能规范定义了一个计算机系统的外部规约，即系统与外部（包括过程、环境、人等）的接口关系，那么，系统的总体设计方案则要确立系统的内部规约，包括系统采用的技术途径、结构，应将任务及其指标分解到系统的各要素之中，如软件、硬件、人，等等。通常一个计算机系统方案应包括以下几个方面的内容：

（1）任务的来源，技术要求，质量指标和经费。主要是任务与经费，要制定项目的实施计划；对人员进行组织、分工；按照计划的进度以及成本、风险、质量的要求，进行系统开发。

（2）采取的技术途径。这里指系统采用的设计水平，究竟是全面自己设计，还是应用一个成熟的系统。对于后者，开发过程要简单的多。这时系统的主要有功能设计和应用软件的开发。硬件和支撑软件往往是成熟的产品，无需再做修改，只是配置问题。

（3）方案的规模，子系统的划分。确立系统的技术途径之后，就要针对采用的技术途径，对照系统功能规范的要求和经费要求逐步确立系统的结构，进行功能分解，设计子系统，并确立各分系统的技术指标，以及各子系统的接口关系。

（4）系统的数据库结构。对于小型的或专用的计算机系统，数据结构较为简单，对于一个通用的系统或大、中型的系统，定义系统的数据结构是很有必要的。

（5）预计系统的各项指标。在此要综合系统及各模块的功能，以检查系统的方案是否可以达到系统功能规范的要求。

（6）所需的设备、仪器、关键元器件、工艺工具等的购置情况及保证条件。

（7）研制周期、异常情况的处理等等。

图3.2显示了计算机系统开发的几个主要步骤。

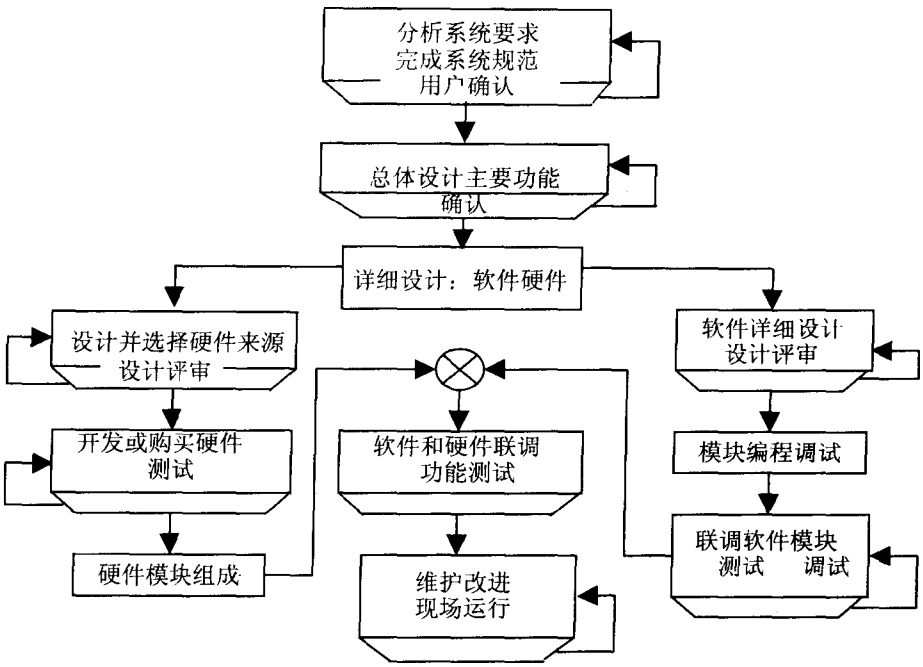

图 3.2 系统开发设计周期

采用不同的技术途径和不同的系统规模，系统的总体方案的内容有很大差别。多数场合下，一个功能可用多种方法实现。系统工程师必须善于根据系统设计目标和约束条件设计并选择最佳方案。

一个计算机系统工程的总体设计方案应包括以下几方面的内容：硬件工程、软件工程、人机工程和数据库工程，本节就此进行阐述。

3.1.2 硬件和硬件工程

基于计算机的系统离不开计算机硬件的支撑。

计算机系统工程师根据系统需求为硬件系统指派任务，产生硬件需求。硬件工程师根据硬件需求设计、制造或选择硬部件或设备，如主机、通用或专用外部设备、网络与通信设备等。

为了建造一个高质量的、用户满意的硬件系统，硬件工程师在考虑硬件系统功能和性能