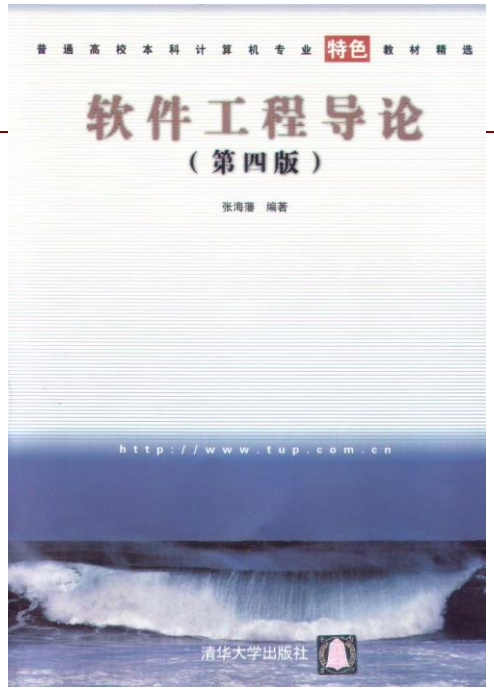
普通高校本科计算机专业特色教材精选

张海藩　编著

软件工程导论（第4版）



总 目 录

总 目 录

第1章 软件工程学概述

迄今为止，计算机系统已经经历了4个不同的发展阶段，但是，我们仍然没有彻底摆脱“软件危机”的困扰，软件已经成为限制计算机系统发展的瓶颈。

为了更有效地开发与维护软件，软件工作者在20世纪60年代后期开始认真研究消除软件危机的途径，从而逐渐形成了一门新兴的工程学科——计算机软件工程学(通常简称为软件工程)。

1.1 软件危机

硬件与软件的区别

什么是软件

“软件作坊” 加剧了软件危机的出现

开发软件就是去编写程序；

接到一个项目马上开始写程序；

为了追求速度而不写或者少写文档；

代码不需要注释；

越早开始写程序，完成整个软件开发所用的时间肯定就越短；

程序在用户处运行时出现错误，开发人员过去修改程序，改好后就运行。

“软件作坊”仍然沿用早期形成的个体化软件开发方法。在程序运行时发现的错误必须设法改正；用户有了新的需求时必须相应地修改程序；硬件或操作系统更新时，通常需要修改程序以适应新的环境。

“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。

1.1.1 软件危机的介绍

软件危机是指在计算机软件的开发和维护过程中所遇到的一系列严重问题。

实际上，几乎所有软件都不同程度地存在这些问题。

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。

(1) 对软件开发成本和进度的估计常常很不准确。为了赶进度和节约成本所采取的一些权宜之计又往往损害了软件产品的质量，从而不可避免地会引起用户的不满。

(2) 用户对“已完成的”软件系统不满意的现象经常发生。

软件开发人员和用户之间的信息交流往往很不充分，有行业盲区，常常导致最终产品不符合用户的实际需要。

(3) 软件产品的质量往往靠不住。

软件的特性决定了软件的可靠性和质量保证是一个模糊的概念。

(4) 软件常常是不可维护的。

很多程序中的错误是非常难改正的，不太可能使这些程序适应新的硬件环境。

(5) 软件通常没有适当的文档资料。

软件不仅仅是程序，还应该有一整套文档资料。对于维护人员而言，这些文档资料更是必不可少的。缺乏必要的文档资料或者文档资料不合格，必然给软件开发和维护带来许多严重的困难和问题。

(6) 软件成本在计算机系统总成本中所占的比例逐年上升。

硬件成本逐年下降，而软件成本却持续上升。

(7) 软件开发生产率提高的速度，远远跟不上计算机应用迅速普及深入的趋势。

软件产品“供不应求”的现象使人类不能充分利用现代计算机硬件提供的巨大潜力。

1.1.2 产生软件危机的原因

在软件开发和维护的过程中存在这么多严重问题，一方面与软件本身的特点有关，另一方面也和软件开发与维护的方法不正确有关。

软件的逻辑特性决定了软件的开发、管理和控制过程都相当困难。软件的维护过程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。

软件经常需要由多人分工合作，然而，如何保证每个人完成的工作合在一起确实能构成一个高质量的大型软件系统，更是一个极端复杂困难的问题，不仅涉及许多技术问题，诸如分析方法、设计方法、形式说明方法、版本控制等，更重要的是必须有严格而科学的管理。

目前很多软件人员对软件开发和维护有不少糊涂观念，在实践过程中或多或少地采用了错误的方法和技术，认为软件开发就是写程序并设法使之运行，轻视软件维护等，这可能是使软件问题发展成软件危机的主要原因。

有的时候，为了赶进度，也可能采用不规范的开发方法，这种错误的作法也导致了软件危机的加剧。

没有认识到软件开发有自己本身的规律，没有循序渐进，按照软件开发的规律办事。

例如：没有成分理解需求就匆忙编写程序是许多软件开发工程失败的主要原因之一。软件开发人员需要做大量深入细致的调查研究工作，反复多次地和用户交流信息，才能真正全面、准确、具体地了解用户的要求。

事实上，越早开始写程序，完成它所需要用的时间往往越长。

一个软件从始到终，要经历一定时期，通常把软件经历的这个时期称为软件生命周期。

软件开发最初的工作是问题定义，也就是确定要解决的问题是什么；

然后要进行可行性研究，决定该问题是否存在一个可行的解决办法；

接下来进行需求分析，也就是深入具体地了解用户的要求，在所要开发的系统必须做什么这个问题上和用户取得完全一致的看法。

接下来才能进入开发时期，在开发时期首先需要对软件进行设计，然后才能进入编写程序的阶段。

然后是程序的测试期。程序编写完之后还必须经过若干阶段的测试工作，才能最终交付使用。所以，编写程序只是软件开发过程中的一个阶段，而且在典型的软件开发工程中，编写程序所需的工作量只占软件开发全部工作量的10%～20%。

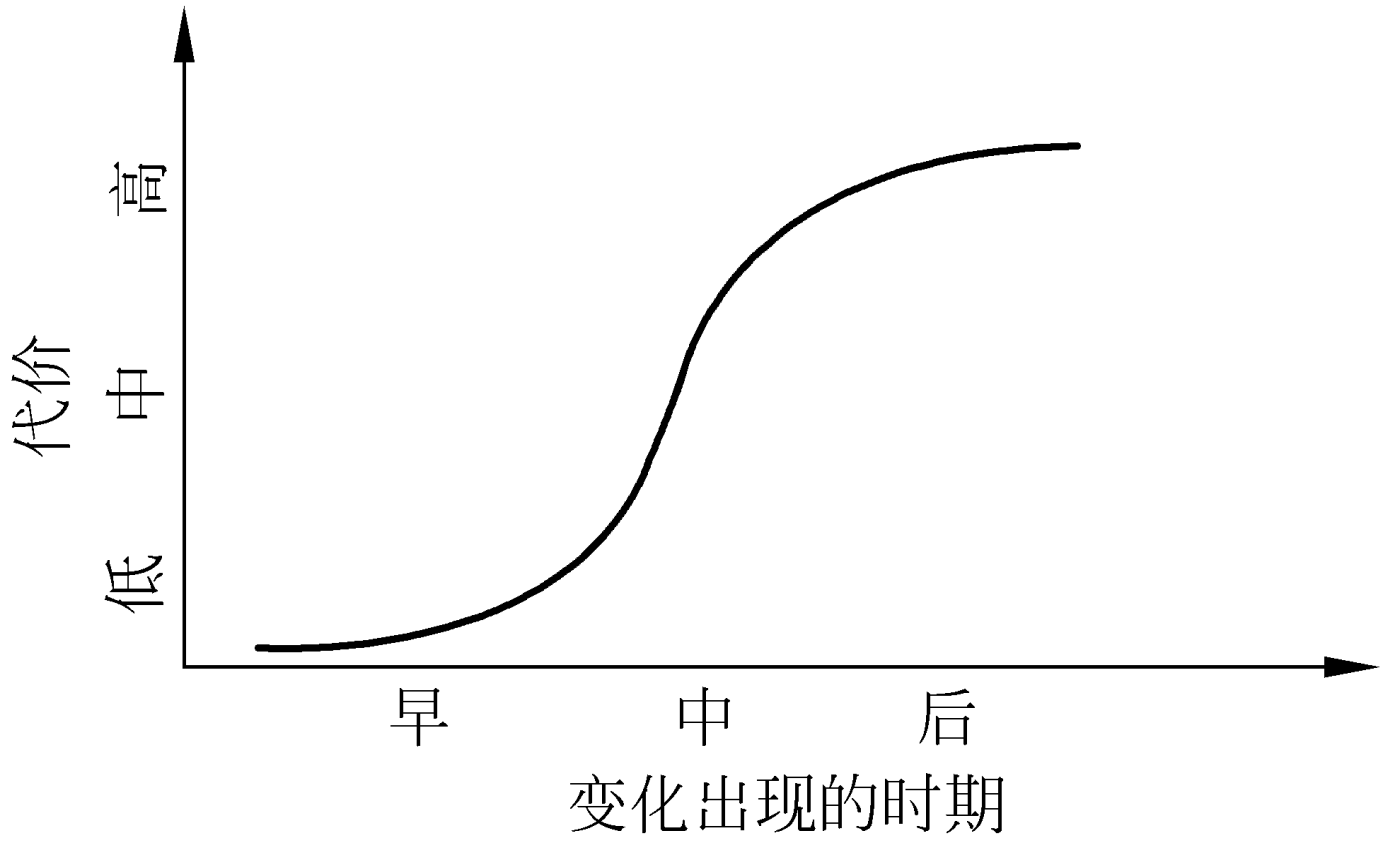


图1.1 引入同一变动付出的代价随时间变化的趋势

所以说，不能轻视软件维护工作。在软件维护期内，不仅必须改正使用过程中发现的每一个潜伏的错误，而且当环境变化时还必须相应地修改软件以适应新的环境。因此维护是极端艰巨复杂的工作，需要花费很大代价。

软件工程的一个重要目标就是提高软件的可维护性，减少软件维护的代价。

1.1.3 消除软件危机的途径

IEEE对软件下的定义：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程序时所必需的数据。其中，方法和规则通常是在文档中说明并在程序中实现的。

一个软件必须由一个完整的配置组成，事实上，软件是程序、数据及相关文档的完整集合。其中，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处理信息的数据结构；文档是开发、使用和维护程序所需要的图文资料。

更重要的是，必须充分认识到软件开发是一种组织良好、管理严密、各类人员协同配合、共同完成的工程项目。必须充分吸取和借鉴各种工程项目所积累的原理、概念、技术和方法，特别要吸取几十年来人类从事计算机硬件研究和开发的经验教训。

总之，为了解决软件危机，既要有技术措施(方法和工具)，又要有必要的组织管理措施。软件工程正是从管理和技术两方面研究如何更好地开发和维护计算机软件的一门新兴学科。

1.2 软件工程

1.2.1 软件工程的介绍

NATO在1968年给出了软件工程的一个定义：“软件工程就是为了经济地获得可靠的且能在实际机器上有效地运行的软件，而建立和使用完善的工程原理。”

IEEE 在1993年给出了一个更全面更具体的定义：“软件工程是：①把系统的、规范的、可度量的途径应用于软件开发、运行和维护过程，也就是把工程应用于软件；②研究①中提到的途径。”

软件工程的本质特性：

1. 软件工程关注于大型程序的构造

“大”与“小”的概念是相对的。

事实上，在此处使用术语“程序”并不十分恰当，现在的软件开发项目通常构造出包含若干个相关程序的“系统”。

2. 软件工程的中心课题是控制复杂性

通常，软件所要解决的问题十分复杂，需要把问题分解，使得分解出的每个部分是简单的、可理解的，而且各部分之间保持简单的通信关系。

用这种方法并不能降低问题的整体复杂性，但是却可使它变成可以管理的。

3. 软件经常变化

绝大多数软件都模拟了现实世界的某一部分。软件为了不被很快淘汰，必须随着现实世界一起变化。

4. 开发软件的效率非常重要

目前，软件供不应求的现象日益严重。所以，软件工程的一个重要课题就是，寻求开发与维护软件的更好更有效的方法和工具。

5. 和谐地合作是开发软件的关键

软件需要多人协同工作才能解决顺利开发。为了有效地合作，必须明确规定每个人的责任和相互通信的方法。为了使大家遵守规定，应该运用标准和规程。通常，可以用工具来支持这些标准和规程。总之，纪律是成功地完成软件开发项目的一个关键。

6. 软件必须有效地支持它的用户

有效地支持用户意味着必须仔细地研究用户，以确定适当的功能需求、可用性要求及其他质量要求(例如，可靠性、响应时间等)。

有效地支持用户还意味着，软件开发不仅应该提交软件产品，而且应该写出用户手册和培训材料。

7. 在软件工程领域中是由具有一种文化背景的人替具有另一种文化背景的人

1.2.2 软件工程的基本原理

软件工作者提出了软件工程的7条基本原理。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

统计显示，在不成功的软件项目中有一半左右是由于计划不周造成的，可见把建立完善的计划作为第一条基本原理是吸取了前人的教训而提出来的。

应该把软件生命周期划分成若干个阶段，并相应地制定出切实可行的计划，然后严格按照计划对软件的开发与维护工作进行管理。

绝不能受客户或上级人员的影响而擅自背离预定计划。

2. 坚持进行阶段评审

软件的质量保证工作贯穿软件开发的始终。因此，在每个阶段都进行严格的评审，以便尽早发现在软件开发过程中所犯的错误，是一条必须遵循的重要原则。

3. 实行严格的产品控制

当需求改变时，为了保持软件各个配置成分的一致性，必须实行严格的产品控制，其中主要是实行基线配置管理。

基线配置管理也称为变动控制：一切有关修改软件的建议，特别是涉及到对基准配置的修改建议，都必须按照严格的规程进行评审，获得批准以后才能实施修改。

4. 采用现代程序设计技术

采用先进的技术不仅可以提高软件开发和维护的效率，而且可以提高软件产品的质量。

5. 结果应能清楚地审查

软件产品是逻辑产品。软件开发人员的工作进展情况可见性差，难以准确度量，从而使得软件产品的开发过程比一般产品的开发过程更难于评价和管理。为了提高软件开发过程的可见性，更好地进行管理，应该根据软件开发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，从而使得所得到的结果能够清楚地审查。

6. 开发小组的人员应该少而精

开发小组人员的素质和数量是影响软件产品质量和开发效率的重要因素。

软件开发小组的组成人员的素质应该好，而人数则不宜过多。

7. 承认不断改进软件工程实践的必要性

遵循上述6条基本原理，就能够按照当代软件工程基本原理实现软件的工程化生产，但是，仅有上述6条原理并不能保证软件开发与维护的过程能跟上技术的不断进步。因此，Boehm提出软件工程的第7条基本原理。按照这条原理，不仅要积极主动地采纳新的软件技术，而且要注意不断总结经验。

1.2.3 软件工程方法学

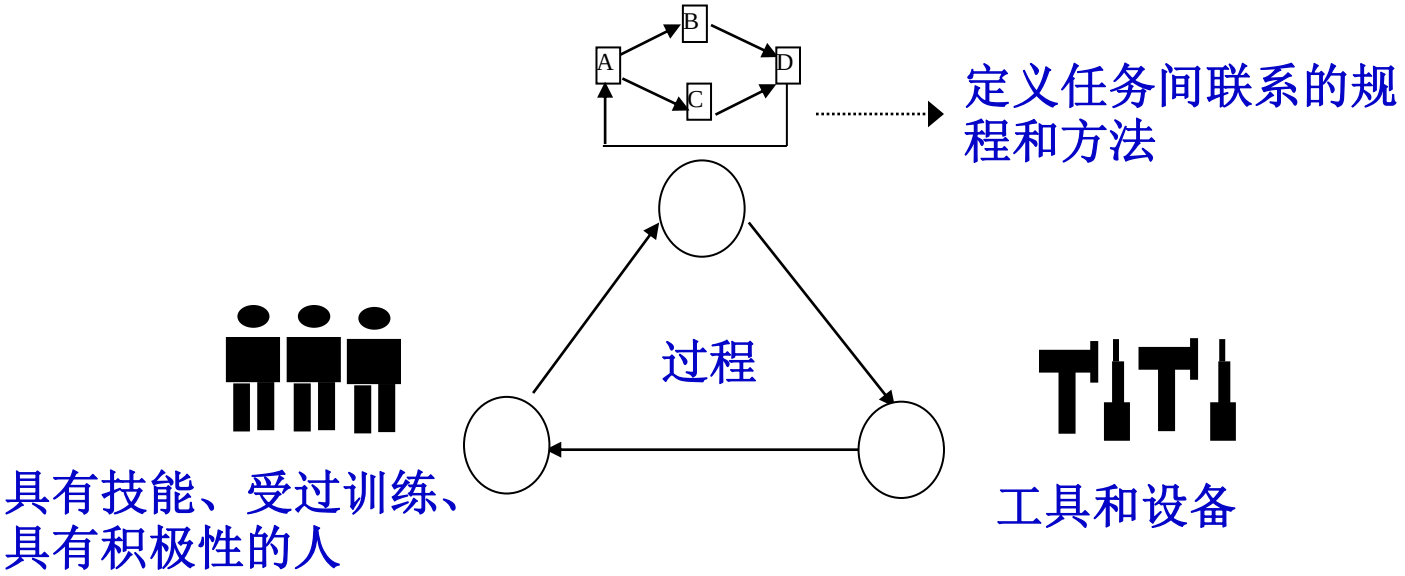
软件工程包括技术和管理两方面的内容，是技术与管理紧密结合所形成的工程学科。

所谓管理就是通过计划、组织和控制等一系列活动，合理地配置和使用各种资源，以达到既定目标的过程。

通常把在软件生命周期全过程中使用的一整套技术方法的集合称为方法学(methodology)，也称为范型(paradigm)。

软件工程方法学包含3个要素：方法、工具和过程。其中，方法是完成软件开发的各项任务的技术方法，回答“怎样做”的问题；工具是为运用方法而提供的自动的或半自动的软件工程支撑环境；过程是为了获得高质量的软件所需要完成的一系列任务的框架，它规定了完成各项任务的工作步骤。

目前使用得最广泛的软件工程方法学，分别是传统方法学和面向对象方法学。



过程将各部分连在一起

1. 传统方法学

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划分为若干个阶段，然后顺序地完成每个阶段的任务。采用这种方法学开发软件的时候，从对问题的抽象逻辑分析开始，一个阶段一个阶段地进行开发。

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。

审查的主要标准就是每个阶段都应该交出最新的文档资料。

采用生命周期方法学可以大大提高软件开发的成功率，软件开发的生产率也能明显提高。

2. 面向对象方法学

与传统方法不同，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起来的方法。

概括地说，面向对象方法学具有下述4个要点。

(1) 把对象(object)作为融合了数据及其操作行为的统一软件构件。用对象分解取代了传统方法的功能分解。

(2) 把所有对象都划分成类(class)。类是对具有相同数据和相同操作的一组相似对象的定义，每个类都定义了一组数据和一组操作。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) 继承。

(4) 封装性。

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，保证了在各项开发活动之间的平滑过渡。

最终的产品由许多独立的对象组成，降低了软件产品的复杂性，提高了软件的可理解性，简化了软件的开发和维护工作。对象可重复使用，对象具有继承性和多态性，进一步提高了面向对象软件的可重用性。

1.3 软件生命周期

软件生命周期由软件定义、软件开发和软件维护3个时期组成，每个时期又划分成若干个阶段。

软件定义时期的任务是： 确定软件开发必须完成的总目标；确定工程的可行性；导出实现工程目标应该采用的策略及系统必须完成的功能；估计完成该项工程需要的资源和成本，并且制定工程进度表。这个时期通常又称为系统分析，由系统分析员负责完成。

软件定义时期通常划分成3个阶段，即：问题定义、可行性研究和需求分析。

开发时期具体设计和实现在前一个时期定义的软件，通常由4个阶段组成：总体设计，详细设计，编码和单元测试，综合测试。其中前两个阶段称为系统设计，后两个阶段称为系统实现。

维护时期的主要任务是使软件持久地满足用户的需要。具体地说，当软件在使用过程中发现错误时应该加以改正；当环境改变时应该修改软件以适应新的环境；当用户有新要求时应该及时改进软件以满足用户的新需要。

1. 问题定义

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?”在实践中它可能是最容易被忽略的一个步骤。

通过对客户的访问调查，系统分析员扼要地写出关于问题性质、工程目标和工程规模的书面报告，经过讨论和必要的修改之后这份报告应该得到客户的确认。

2. 可行性研究

这个阶段要回答的关键问题是：“对于上一个阶段所确定的问题有行得通的解决办法吗?”

为了解决这个问题，系统分析员需要进行一次头脑中的系统分析和设计过程。

可行性研究过程比较简短，这个阶段的任务不是具体解决问题，而是研究问题是否值得去解，是否有可行的解决办法。

可行性研究的结果是使用部门负责人作出是否继续进行这项工程的决定的重要依据。

3. 需求分析

这个阶段的任务是准确地确定“为了解决这个问题，目标系统必须做什么”，主要是确定目标系统必须具备哪些功能。

用户的长处和困难

开发人员的长处和困难

因此，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。

系统逻辑模型是设计和实现目标系统的基础，因此必须准确完整地体现用户的要求。需要用文档准确地记录对目标系统的需求，这份文档通常称为规格说明书(specification)。

4. 总体设计

这个阶段必须回答的关键问题是：“概括地说，应该怎样实现目标系统?”总体设计又称为概要设计。

首先，应该设计出实现目标系统的几种可能的方案。软件工程师应该用适当的表达工具描述每种方案，分析每种方案的优缺点，并在充分权衡各种方案的利弊的基础上，推荐一个最佳方案。此外，还应该制定出实现最佳方案的详细计划。如果客户接受所推荐的方案，则应该进一步完成下述的另一项主要任务。

设计工作确定了解决问题的策略及目标系统中应包含的程序。总体设计的另一项主要任务就是设计程序的体系结构，也就是确定程序由哪些模块组成以及模块间的关系。

5. 详细设计

详细设计阶段的任务就是把解法具体化，也就是回答下面这个关键问题：“应该怎样具体地实现这个系统呢?”

这个阶段的任务还不是编写程序，而是设计出程序的详细规格说明。程序员可以根据它们写出实际的程序代码。

详细设计也称为模块设计，在这个阶段将详细地设计每个模块，确定实现模块功能所需要的算法和数据结构。

6. 编码和单元测试

这个阶段的关键任务是写出正确的容易理解、容易维护的程序模块。

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，把详细设计的结果翻译成用选定的语言书写的程序，并且测试每一个模块。

7. 综合测试

这个阶段的关键任务是通过各种类型的测试使软件达到预定的要求。

最基本的测试是集成测试和验收测试。

集成测试是根据设计的软件结构，把经过单元测试检验的模块按某种选定的策略装配起来，在装配过程中对程序进行必要的测试。

验收测试则是按照规格说明书的规定，由用户对目标系统进行验收。

必要时还可以再通过现场测试或平行运行等方法对目标系统进一步测试检验。

7. 综合测试

为了使用户能够积极参加验收测试，并且在系统投入运行以后能够正确有效地使用这个系统，通常需要对用户进行培训。

通过对软件测试结果的分析可以预测软件的可靠性；反之，根据对软件可靠性的要求，也可以决定测试和调试过程什么时候可以结束。

同时用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。

8. 软件维护

维护阶段的关键任务是，通过各种必要的维护活动使系统持久地满足用户的需要。

通常有四类维护活动：

改正性维护：诊断和改正在使用过程中发现的软件错误；

适应性维护：修改软件以适应环境的变化；

完善性维护：根据用户的要求改进或扩充软件使它更完善；

预防性维护：修改软件为将来的维护活动预先做准备。

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。

在实际从事软件开发工作时，软件规模、种类、开发环境及开发时使用的技术方法等因素，都影响阶段的划分。

1.4 软件过程

软件过程是为了获得高质量软件所需要完成的一系列任务的框架，它规定了完成各项任务的工作步骤。

在完成开发任务时必须进行一些开发活动，并且使用适当的资源，在过程结束时将把输入转化为输出。因此，ISO 9000把过程定义为：使用资源将输入转化为输出的活动所构成的系统。

“系统”的含义是广义的：“系统是相互关联或相互作用的一组要素。”

过程定义了运用方法的顺序、应该交付的文档资料、为保证软件质量和协调变化所需要采取的管理措施，以及标志软件开发各个阶段任务完成的里程碑。

没有一个适用于所有软件项目的任务集合。通常，一个任务集合包括一组软件工程任务、里程碑和应该交付的产品。

使用生命周期模型描述软件过程。生命周期模型规定了把生命周期划分成哪些阶段及各个阶段的执行顺序，也称为过程模型。

1.4.1 瀑布模型

瀑布模型是应用得最早，也是最广泛的生命周期模型。传统软件工程方法学的软件过程，基本上可以用瀑布模型来描述。

图1.2所示为传统的瀑布模型。按照传统的瀑布模型开发软件，有下述的几个特点。

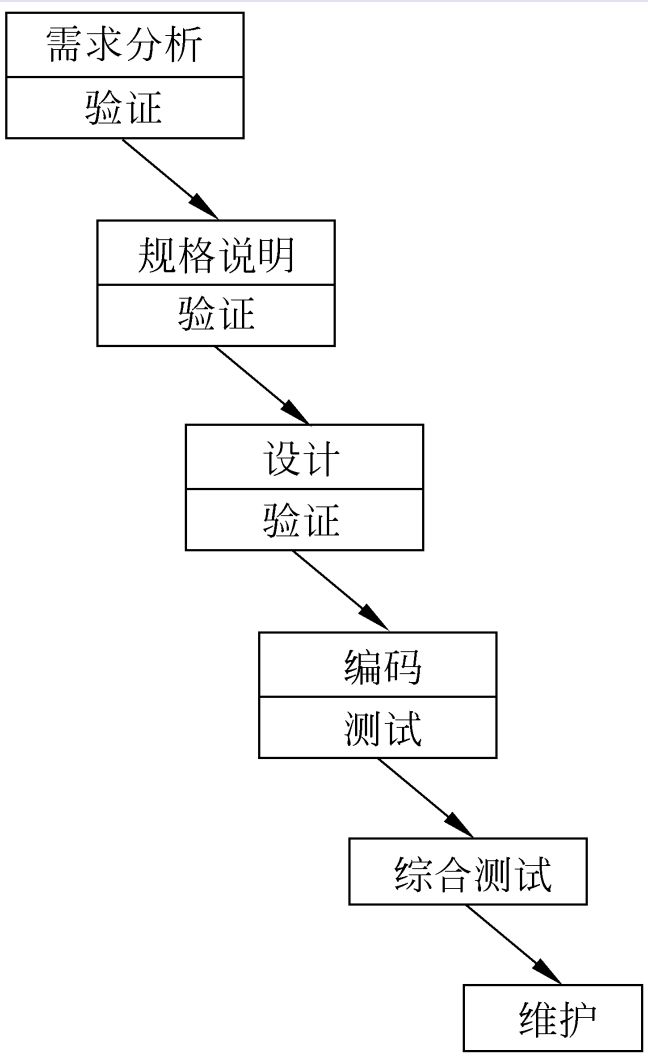


图1.2 传统的瀑布模型

1. 阶段间具有顺序性和依赖性

这个特点有两重含义：①必须等前一阶段的工作完成之后，才能开始后一阶段的工作；②前一阶段的输出文档就是后一阶段的输入文档，因此，只有前一阶段的输出文档正确，后一阶段的工作才能获得正确的结果。

2. 推迟实现的观点

对于规模较大的软件项目来说，往往编码开始得越早最终完成开发工作所需要的时间反而越长。这是因为，前面阶段的工作没做或做得不扎实，过早地考虑程序实现，往往导致大量返工，有时甚至发生无法弥补的问题。

瀑布模型在编码之前设置了系统分析与系统设计的各个阶段，分析与设计阶段的基本任务规定，在这两个阶段主要考虑目标系统的逻辑模型，不涉及软件的物理实现。

清楚地区分逻辑设计与物理设计，尽可能推迟程序的物理实现，是按照瀑布模型开发软件的一条重要的指导思想。

3. 质量保证的观点

软件工程的基本目标是优质、高产。为了保证所开发的软件的质量，在瀑布模型的每个阶段都应坚持两个重要做法：

（1） 每个阶段都必须完成规定的文档，没有交出合格的文档就是没有完成该阶段的任务。完整、准确的合格文档不仅是软件开发时期各类人员之间相互通信的媒介，也是运行时期对软件进行维护的重要依据。

（2） 每个阶段结束前都要对所完成的文档进行评审，以便尽早发现问题，改正错误。及时审查，是保证软件质量，降低软件成本的重要措施。

实际的瀑布模型是带“反馈环”的，如图1.3所示。当在后面阶段发现前面阶段的错误时，需要沿图中左侧的反馈线返回前面的阶段，修正前面阶段的产品之后再回来继续完成后面阶段的任务。

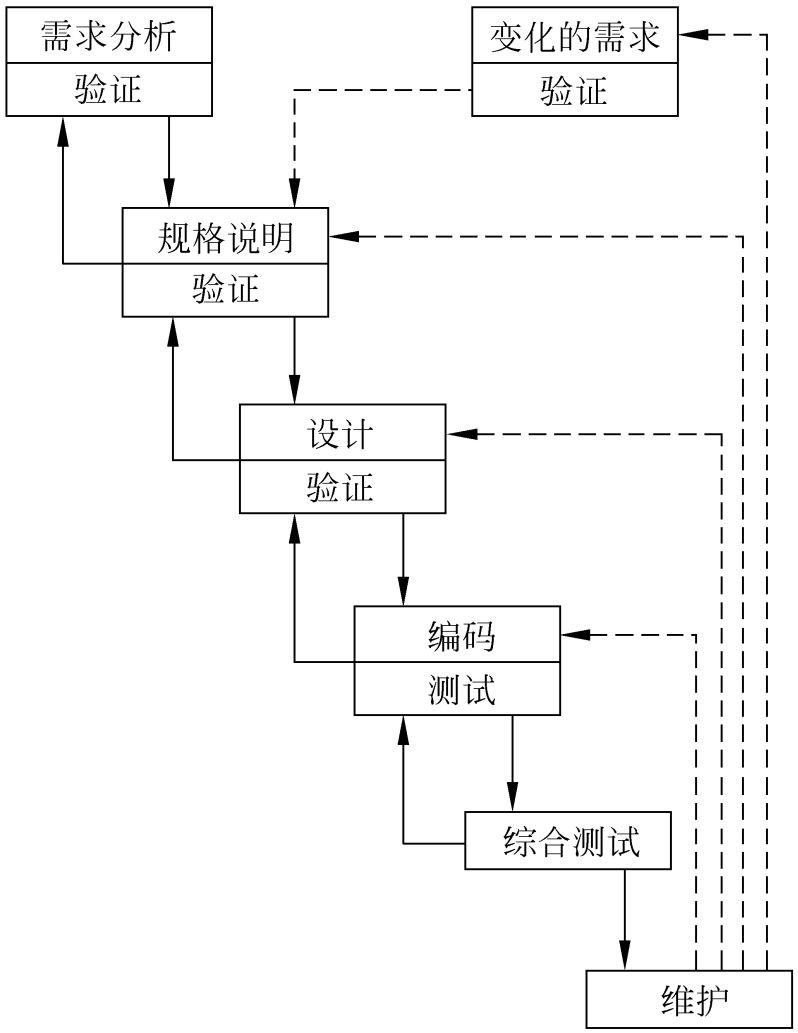


图1.3 实际的瀑布模型

瀑布模型有许多优点：可迫使开发人员采用规范的方法； 严格地规定了每个阶段必须提交的文档；要求每个阶段交出的所有产品都必须经过质量保证小组的验证。

各个阶段产生的文档是维护软件产品时必不可少的，没有文档的软件几乎是不可能维护的。遵守瀑布模型的文档约束，将使软件维护变得比较容易一些，能显著降低软件预算。

可以说，瀑布模型的成功在很大程度上是由于它基本上是一种文档驱动的模型。

在软件产品交付之前，用户只能通过文档来了解产品是什么样的。事实上，要求用户不经过实践就提出完整准确的需求，在许多情况下是不切实际的。总之，由于瀑布模型几乎完全依赖于书面的规格说明，很可能导致最终开发出的软件产品不能真正满足用户的需要。

采用瀑布模型进行软件开发，往往需要较长的开发周期。

1.4.2 快速原型模型

快速原型是快速建立起来的可以在计算机上运行的程序，它所能完成的功能往往是最终产品能完成的功能的一个子集。如图1.4所示，快速原型模型的第一步是快速建立一个能反映用户主要需求的原型系统，让用户在计算机上试用它，通过实践来了解目标系统的概貌。

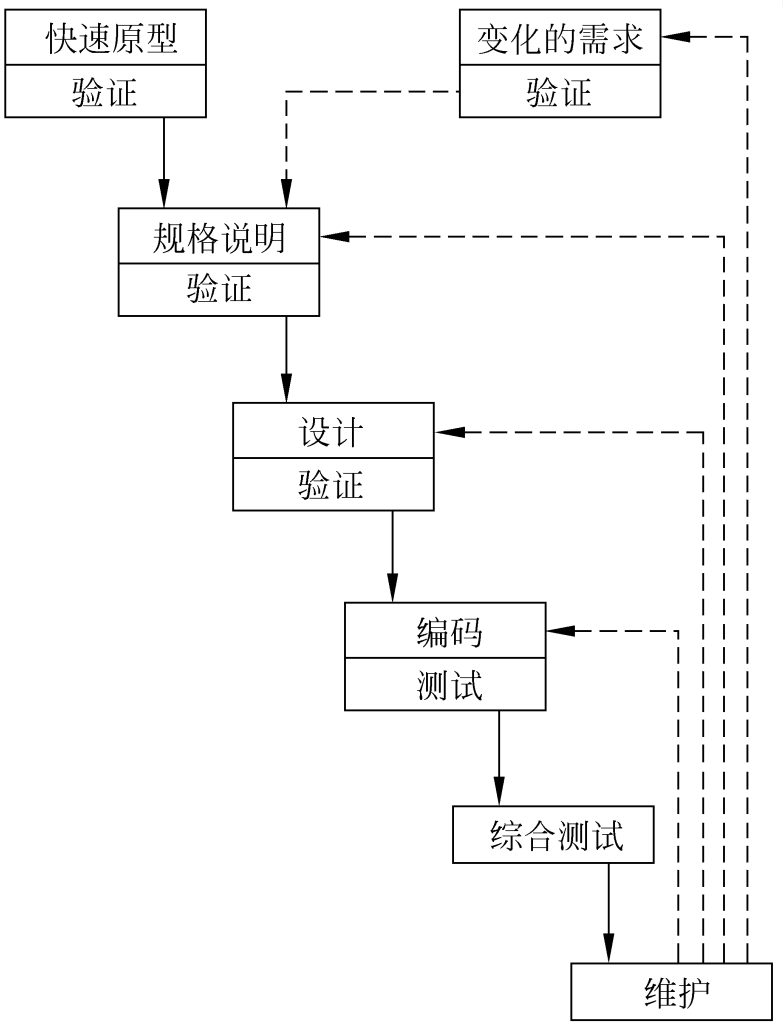


图1.4 快速原型模型

通常，用户试用原型系统之后会提出许多修改意见，开发人员按照用户的意见快速地修改原型系统，然后再次请用户试用……一旦用户认为这个原型系统确实能做他们所需要的工作，开发人员便可据此书写规格说明文档，根据这份文档开发出的软件可以满足用户的真实需求。

原型模型是不带反馈环的， 软件产品的开发基本上是线性顺序进行的。能做到基本上线性顺序开发的主要原因如下：

（1） 原型系统已经通过与用户交互而得到验证，据此产生的规格说明文档正确地描述了用户需求，因此，在开发过程的后续阶段不会因为发现了规格说明文档的错误而进行较大的返工。

（2） 开发人员通过建立原型系统已经了解许多，因此，在设计和编码阶段发生错误的可能性也比较小。

快速原型的本质是“快速”。开发人员应该尽可能快地建造出原型系统，以加速软件开发过程，节约软件开发成本。

原型的用途是获知用户的真正需求，一旦需求确定了，原型将被抛弃。

1.4.3 增量模型

增量模型也称为渐增模型，如图1.5所示。使用增量模型开发软件时，把软件产品作为一系列的增量构件来设计、编码、集成和测试。每个构件由多个相互作用的模块构成，并且能够完成特定的功能。使用增量模型时，第一个增量构件往往实现软件的基本需求，提供最核心的功能。第二个增量构件提供更完善的编辑和文档生成功能；第三个增量构件实现拼写和语法检查功能；第四个增量构件完成高级的页面排版功能。

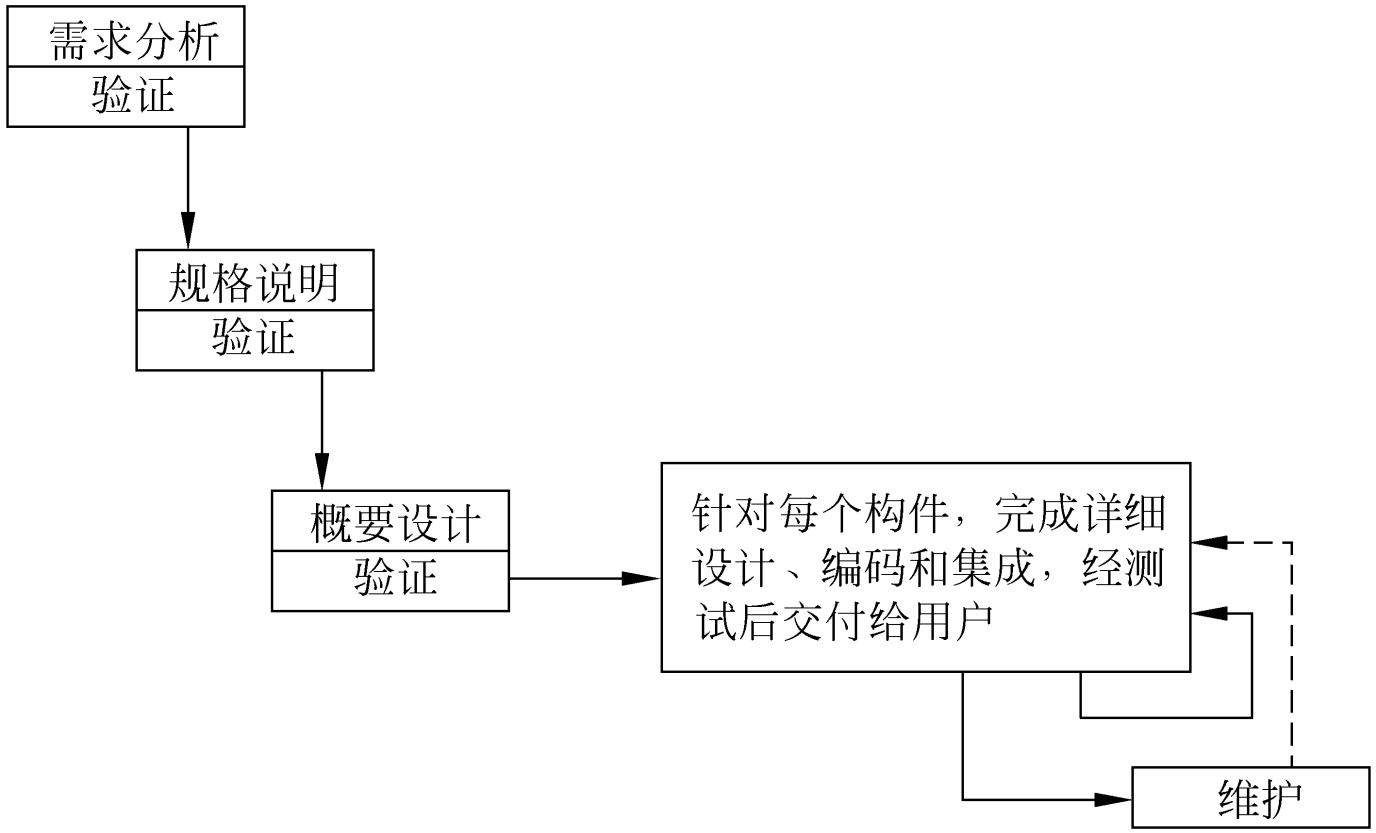


图1.5 增量模型

把软件产品分解成增量构件时，应该使构件的规模适中，规模过大或过小都不好。

采用瀑布模型或快速原型模型开发软件时，目标都是一次就把一个满足所有需求的产品提交给用户。增量模型则与之相反，它分批地逐步向用户提交产品，整个软件产品被分解成许多个增量构件，开发人员一个构件接一个构件地向用户提交产品。从第一个构件交付之日起，用户就能做一些有用的工作。能在较短时间内向用户提交可完成部分工作的产品，是增量模型的一个优点。

增量模型的另一个优点是，逐步增加产品功能可以使用户有较充裕的时间学习和适应新产品，从而减少一个全新的软件可能给客户组织带来的冲击。

使用增量模型的困难是，在把每个新的增量构件集成到现有软件体系结构中时，必须不破坏原来已经开发出的产品。必须把软件的体系结构设计得便于按这种方式进行扩充，向现有产品中加入新构件的过程必须简单、方便，软件体系结构必须是开放的。从长远观点看，具有开放结构的软件拥有真正的优势，这样的软件的可维护性明显好于封闭结构的软件。

1.4.4 螺旋模型

软件风险是任何软件开发项目中都普遍存在的实际问题，项目越大，承担该项目所冒的风险也越大。软件风险可能在不同程度上损害软件开发过程和软件产品质量。因此，在软件开发过程中必须及时识别和分析风险，并且采取适当措施以消除或减少风险的危害。

构建原型是一种能使某些类型的风险降至最低的方法。螺旋模型的基本思想是，使用原型及其他方法来尽量降低风险。

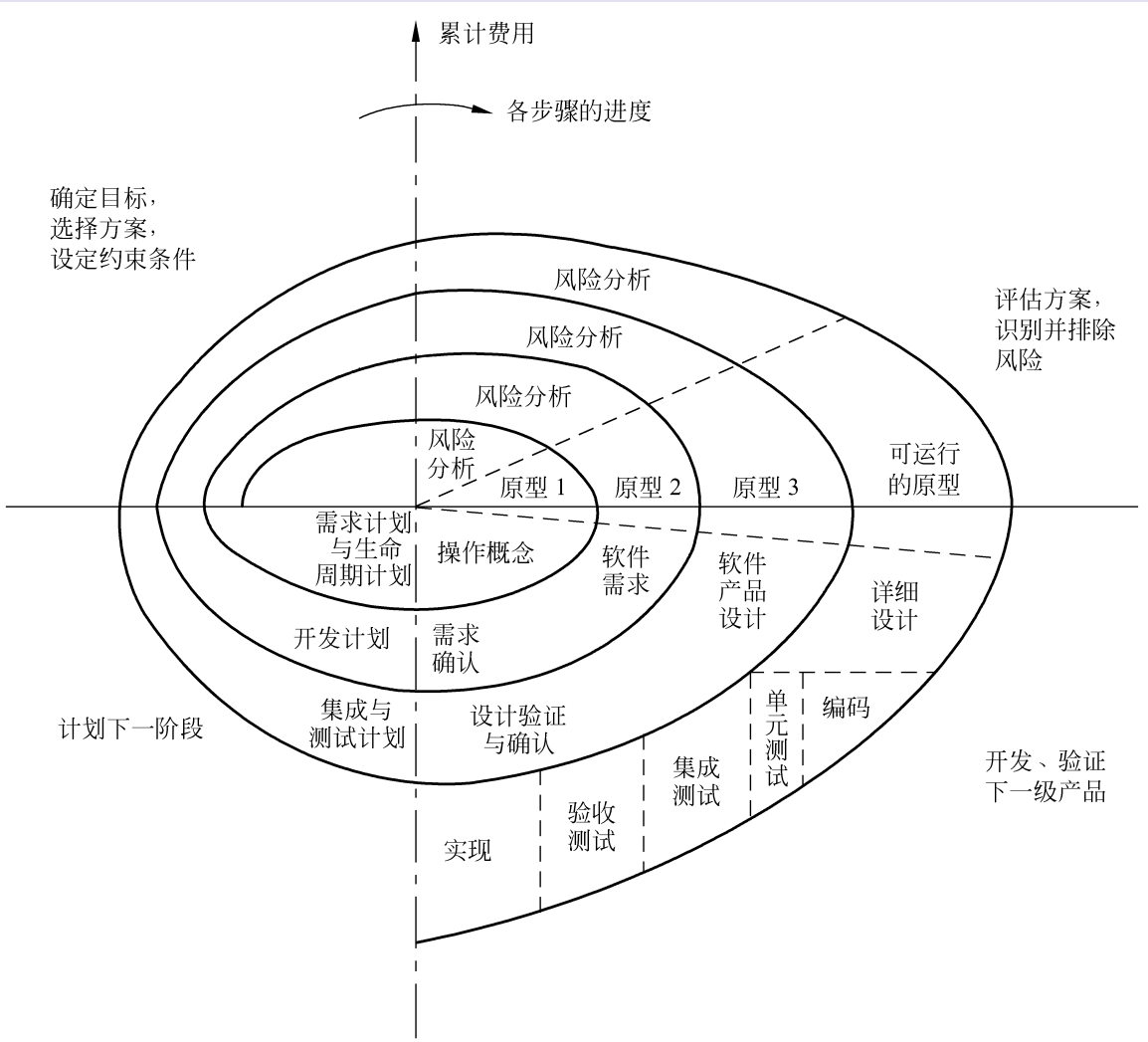

累计费用
各步骤的进度
确定目标，
选择方案，
设定约束条件
风险分析
风险分析
风险分析
风险
分析
评估方案，
识别并排除
风险
原型 1
原型 2
原型 3
可运行
的原型
需求计划
与生命
周期计划
操作概念
软件
需求
软件
产品
设计
详细
设计
开发计划
需求
确认
集成与
测试计划
设计验证
与确认
单元测试
编码
计划下一阶段
集成
测试
开发、验证
下一级产品
验收
测试
实现

图1.8 完整的螺旋模型

螺旋模型有许多优点：对可选方案和约束条件的强调有利于已有软件的重用，也有助于把软件质量作为软件开发的一个重要目标；减少了过多测试（浪费资金）或测试不足（产品故障多）所带来的风险。

螺旋模型主要适用于内部开发的大规模软件项目。如果进行风险分析的费用接近整个项目的经费预算，则风险分析是不可行的。事实上，项目越大，风险也越大，因此，进行风险分析的必要性也越大。

1.5 小结

本章首先介绍了软件危机产生的原因及其现象，然后介绍了软件工程的概念，并用生命周期方法学把软件生命周期划分为若干个相对独立的阶段，每个阶段完成一些确定的任务，交出最终的软件配置的一个或几个成分(文档或程序)； 基本上按顺序完成各个阶段的任务，在完成每个阶段的任务时采用结构化技术和适当的软件工具；在每个阶段结束之前都进行严格的技术审查和管理复审。

把软件生命周期划分成问题定义、可行性研究、需求分析、总体设计、详细设计、编码和单元测试、综合测试以及运行维护等8个阶段。

软件过程是为了获得高质量的软件产品所需要完成的一系列任务的框架，它规定了完成各项任务的工作步骤。由于没有一个适用于所有软件项目的任务集合，科学、有效的软件过程应该定义一组适合于所承担的项目特点的任务集合。

使用软件过程模型简洁地描述软件过程，本章介绍了4种典型的软件过程模型。

瀑布模型历史悠久、广为人知，它的优势在于它是规范的、文档驱动的方法。

快速原型模型通过快速构建起一个可在计算机上运行的原型系统，让用户试用原型并收集用户反馈意见的办法，获取用户的真实需求。

增量模型具有可在软件开发的早期阶段使投资获得明显回报和较易维护的优点，但是，要求软件具有开放的结构是使用这种模型时固有的困难。

风险驱动的螺旋模型适用于内部开发的大型软件项目，但是，只有在开发人员具有风险分析和排除风险的经验及专门知识时，使用这种模型才会获得成功。

习题

1-1 什么是软件危机?它有哪些典型表现?为什么会出现软件危机?

1-2 假设你是一家软件公司的总工程师，当你把图1.1给手下的软件工程师们观看，告诉他们及早发现并改正错误的重要性时，有人不同意你的观点，认为要求在错误进入软件之前就清除它们是不现实的，并举例说：“如果一个故障是编码错误造成的，那么，一个人怎么能在设计阶段清除它呢?”你怎么反驳他?

1-3 什么是软件工程?它有哪些本质特性?怎样用软件工程消除软件危机?

1-4 简述结构化范型和面向对象范型的要点，并分析它们的优缺点。

1-5 根据历史数据可以做出如下的假设：

对计算机存储容量的需求大致按下面公式描述的趋势逐年增加：$M=4080e^{0.28(Y-1960)}$

存储器的价格按下面公式描述的趋势逐年下降：$P1=0.3\times0.72^{Y-1974}$(美分/位)

如果计算机字长为16位，则存储器价格下降的趋势为：$P2=0.048\times0.72^{Y-1974}$(美元/字)

在上列公式中Y代表年份，M是存储容量(字数)，P1和P2代表价格。

基于上述假设可以比较计算机硬件和软件成本的变化趋势。要求计算：

(1) 在1985年对计算机存储容量的需求估计是多少？如果字长为16位，这个存储器的价格是多少？

(2) 假设在1985年一名程序员每天可开发出10条指令，程序员的平均工资是每月4000美元。如果一条指令为一个字长，计算使存储器装满程序所需用的成本。

(3) 假设在1995年存储器字长为32位，一名程序员每天可开发出30条指令，程序员的月平均工资为6000美元，重复(1)、(2)题。

1-6 什么是软件过程?它与软件工程方法学有何关系?

1-7 什么是软件生命周期模型?试比较瀑布模型、快速原型模型、增量模型和螺旋模型的优缺点，说明每种模型的适用范围。