高等学校
电子信息类 规划教材

软件工程基础

SOFTWARE PROJECT BASIC

汤 惟

西安交通大学出版社

高等学校电子信息类规划教材

软件工程基础

汤　惟

西安交通大学出版社
·西安·

内容简介

本书作为大专计算机专业软件工程课程教材，以实用软件开发、维护的基本原理、主要技术为主要内容，力求突出应用主流，强调学以致用和软件工程规范，适当反映软件工程学的最新进展，以达到软件工程从需求分析、软件设计与实现、软件维护以及软件开发项目管理等方面的教学目的。本书也可作为非计算机专业本科生学习软件工程课程和计算机应用工程人员继续教育的培训教材或教学参考书。

图书在版编目(CIP)数据

软件工程基础/汤惟著. -西安:西安交通大学出版社,2000.2
ISBN 7-5605-1162-7

Ⅰ.软… Ⅱ.汤… Ⅲ.软件工程-基本知识
Ⅳ.TP311.5

中国版本图书馆 CIP 数据核字(2000)第 03411 号

*

西安交通大学出版社出版发行
(西安市兴庆南路 25 号 邮政编码:710049 电话:(029)2668315)
陕西宝石兰印务有限责任公司印装
各地新华书店经销

*

开本:787mm×1 092mm 1/16 印张:11.25 字数:267 千字
2000 年 3 月第 1 版 2001 年 8 月第 2 次印刷
印数:5 001~8 000 定价:15.00 元

前　言

本教材系按原电子工业部《1996～2000年全国电子信息类专业教材编审出版规划》的要求，由全国计算机专业大专教学指导委员会编审、推荐出版。本教材由华中理工大学汉口分校汤惟编写，武汉大学数学与计算机科学学院郑慧娆教授担任主审，南京机械高等专科学校李逊林老师担任责任编委。

随着社会信息化的进程，计算机软件产业正在成为知识经济重要组成部分，以软件的定义、开发、维护和管理为内容的软件工程学对于人们已经不是一个陌生的学科。从事计算机软件和应用工程开发的技术人员已经普遍认识到，如果一个项目不遵循软件工程的基本原则，不自觉地使用软件工程方法，不能提交规范的软件文档，必将受到实践的惩罚。软件工程课程作为计算机专业的一门综合性工程应用课程，即为学生后续的专业课程教学和工程实践环节提供必要的方法学基础和能力训练，同时也是学生参加工作后立即面临的最直接应用基础。

软件工程覆盖的范围包括软件开发技术、软件工程环境、软件工程管、软件质量与可靠性控制、软件经济学和软件心理研究等方面内容。近年来面向对象方法学和技术的研究与应用普及，对软件工程产生了深远的影响。从方法学角度看，目前软件工程正处于面向对象技术的过渡期。但是，万变不离其宗，软件工程始终以如何高效地生产高质量、可维护软件产品作为其目标。面向对象的方法也是从结构化方法发展而来的，在软件工程应用的入门阶段，结构化方法仍然是最基本的要求和大多数中、小型软件开发中的实用技术。因此从基础、实用的观点出发，本教材以软件生存期为主线，重点讨论结构化思想为主的软件开发基本方法和技术，包括结构化需求分析、设计、编码和测试，同时也兼顾到它们与面向对象技术的联系与发展。从工程观点出发，软件开发与维护不是纯技术问题，项目管理与组织、软件配置与质量控制也是软件技术人员必须掌握的基本知识，因此教材中也做了适当介绍。

使用本教材的参考讲授学时为30学时。在课程组织方面应该结合软件开发实例进行讲授，使学生通过具体的软件问题求解和工程组织掌握软件工程基本原理和技术。根据这一思想，教材中从第2章开始提出了一个学生成绩管理信息系统开发的完整任务。在教学中，教师应该要求学生对该课题(或者其它类似课题)开展工作，并随教学进程完成相应阶段的任务，按照附录给出的文档参考大纲编制、提交相应文档。此外，还应安排10～30上机学时，组织学生以小组为单位对所布置的课题进行编码实现、测试调试。通过以上教学环节的组织实施，相信学生能够获得比较实际的工程训练，其软件开发技术与组织能力应该有显著的提高。

本教材编写工作中，从选题到教材组织和出版，得到江汉大学乔维声老师的大力支持和帮助，武汉大学郑慧娆教授作为本教材主审，在百忙中对教材内容进行了仔细的审阅，提出了许多宝贵的指导性意见和建议。在此一并表示诚挚的感谢。由于编者的水平有限，书中难免存在缺点和不足，恳请读者批评指正。

编者

1999年12月

软件工程基础

汤 惟

目　录

第 1 章　软件工程概述

1.1　软件工程与软件危机

软件工程学作为指导软件开发与维护实践的理论，是在人们为解决 60 年代开始出现的软件危机中逐步形成和发展起来的。近 30 年来，随着软件产品的系列化，软件开发的工程化、标准化、产业化，软件工程学成为支撑软件产业发展的重要技术理论基础。

1.1.1　软件的发展阶段

自从 20 世纪 40 年代第一台计算机诞生以来，随着计算机科学与工程的发展，计算机软件技术的发展大体可以分为程序设计，软件系统与软件工程三个阶段。

程序设计阶段：60 年代初以前，计算机软件是计算机系统硬件的附属物。那时人们认为软件就是系统或者用户程序。所谓系统程序主要指附着于硬件的输入输出例程库和程序调试器等简单的系统维护工具程序。所谓用户程序通常是为求解某个特定科学、工程计算或者某个具体数据处理要求的单一程序。这些程序规模不大，结构简单，功能单一，许多情况下是谁编制谁使用，程序设计活动高度个人化、技艺化。

软件系统阶段：从 60 年代到 70 年代初，计算机经历了从电子管到晶体管，到集成电路的跃迁；为计算机技术能够在经济、军事发展的关键领域中应用提供了坚实的物质基础。随着硬件技术的发展，计算机软件规模日益增大，其结构、功能越来越复杂。计算机高级语言大量出现；操作系统引入了多道程序、多用户分时与实时处理概念；数据库系统开始出现。大型商业金融数据处理，大型企业生产管理与过程控制，以军备竞赛、空间科学为代表所要求的数据处理、复杂计算、控制系统，这些都要求装备大型复杂的计算机系统。这些计算机系统不仅要求最新的计算机硬件配置，同时也要求“大程序系统”为特征的软件支持。

“大程序系统”需求的急剧增长使软件技术从方法、开发效率等方面远不能适应。随着软件需求的规模、数量剧增和交付要求迫切，“大程序系统设计”已经成为工程项目，其开发过程要求多人参与，分工协作，严密组织。同时程序设计与使用逐步分离，软件开发开始专业化，“大程序设计”再也不是个人的技艺活动了。软件开发的组织要求，应用需求对软件系统质量、可靠性的要求与程序设计个体化，非物化特征形成尖锐矛盾，出现了严重的“软件危机”。整个 60 年代，计算机科学家围绕着程序设计方法，软件开发模型，软件开发支持工具与环境，软件开发方法学开展了范围广泛的研究。为克服软件危机，就若干问题进行了深入讨论并取得如下主要成果：提出了结构化设计方法学并设计了以 Pascal 语言为代表的一批结构化程序设计语言；提出了软件生命周期概念和软件开发的瀑布型模型；明确了文档是软件产品的组成部分，对软件开发各阶段的文档规格进行了初步规范；提出了软件可靠性模型，质量控制的基本概念；开发了一些支持工具软件。1968 年北大西洋公约组织有关计算机科学的一次国际会议上正式使用了“软件工程”这个名词，正式宣告软件工程这一新兴工程学科的诞生。

软件工程阶段:从70年代初至今,随着超大规模集成电路技术的迅猛发展,微型计算机的出现,计算机应用技术已经深入到社会生活的各个领域。在这一阶段,软件开发逐渐专业化。以软件产品系列化,工程化,标准化为特征的软件产业正在发展为21世纪知识经济的支柱产业之一。跨国大型软件公司的出现,软件产品不再依附于硬件,软件工程规范的标准化趋势,软件复用与软件生产管理技术的研究与实践,面向对象方法学的发展等,使得软件工程学在解决软件危机问题的过程中不断地发展、成熟,其主要成果包括实现了以C++为代表的面向对象的语言和软件开发环境系统,计算机辅助软件工程(CASE)的研究与实践,各类具有一定自动化程度CASE产品的出现等。

与现代社会各种计算机应用对于计算机软件的巨大需求相比,目前软件技术进展和软件生产能力远不能满足社会要求。随着软件生产专业化、工厂化进程所产生的新问题,软件危机的许多表现远未得到克服。目前软件工程学从软件系统实现方法学到软件生产管理、质量评估、成本控制等成为一门综合性极强的工程应用学科,其任重道远,发展领域广阔。

1.1.2 · 软件工程与软件危机的产生

20世纪60年代,伴随着计算机系统制造技术的进步,计算机应用对于软件的需求剧增。软件规模功能日益复杂,需求急剧增大。计算机软件开发从早期以个人活动为主的手工作坊方式逐步转到以程序员组形式为代表的集体开发为主。在这一转换过程中,出现了软件生产与市场需求极不适应的严重现象——软件危机。软件危机的主要表现包括:

(1) 软件生产不能满足日益增长的软件需求,软件生产率远低于硬件生产率和计算机应用的增长,出现了软件供不应求的局面。更为严重的是软件生产效率随软件规模的增加和软件复杂性的提高而急剧下降。软件产品的"供不应求"使得计算机硬件的巨大潜力远未发挥。

(2) 软件生产率随软件规模与复杂性提高而下降,智力密集造成的人力成本增加,导致软件成本在计算机系统成本构成中比例急剧上升。早期的计算机系统中软件成本通常不超过系统总成本的20%。到70年代约占40%~60%,80年代中以来,发达国家许多软件产品成本占计算机系统总成本构成的90%以上。

(3) 软件开发进度与成本失控。很难估计软件开发的成本与进度,通常是预算成倍突破,项目计划进度一再延期。软件开发单位为赶进度,控制成本往往只有降低软件质量。软件开发陷入成本居高不下,软件质量无保证,用户不满,开发单位信誉降低的怪圈中。

(4) 软件系统实现的功能与实际需求不符。软件开发人员对用户需求缺乏深入的理解,往往急于编程。闭门造车,最后实现的系统与用户需求相去太远。

(5) 交付的软件难以维护。程序中的错误很难改正,要想使软件适应新的运行环境几乎不可能,软件使用过程中不能增加用户需要的新功能。"可重复使用的软件"还是一个人们努力追求的目标,大量的软件人员在重复开发基本类似的软件。

(6) 软件文档配置没有受到足够的重视。软件文档包括开发过程各阶段的说明书,数据词典,程序清单,软件使用、维护手册,软件测试报告及测试用例。这些软件文档的不规范,不健全是造成软件开发进程、成本不可控制,软件维护、管理、交流困难的重要原因。软件质量缺乏度量依据。

软件危机的表现实际上是软件开发与维护中存在的具有共性的种种问题。近30年来,为解决这些问题,计算机科学家和软件产业从业者已经做出了巨大的努力。但许多问题远未解

决，有些问题的严重性得到部分缓解，新的问题又不断出现。

软件危机产生的原因可以从两个方面加以认识：一是软件产品的固有特性；二是软件专业人员自身的缺陷。

软件的不可见性是软件产品的固有特点之一。与硬件产品不同，软件是计算机系统中的逻辑部件。软件开发过程中，在程序代码运行之前，开发工作的质量、进度难以度量。最终软件产品的使用价值是在软件运行过程中体现出来的。而且软件运行没有“磨损”，这使得软件产品可靠性难以度量，故障隐蔽性强。对原有故障的修改可能导致新的错误。

软件产品的固有特点之二是软件的规模与逻辑复杂性。现代软件产品往往规模庞大，逻辑结构十分复杂。从软件开发管理角度看，软件生产率常随软件规模和复杂性的增大而下降。也就是说，如果一个程序员一年可以开发 1 万条代码，则一个 400 万条代码的软件系统绝对不是集中 400 个人在一年内可以完成的，软件规模与其逻辑复杂度通常并不是线性比关系。当多人合作完成一个系统时，作为一个工程项目，参与人员的组织与信息交流、工作质量与进度控制等更是一个极复杂的问题。就目前的软件技术水平而言，软件开发工作量随软件规模呈几何级数上升。

软件开发人员自身的缺陷主要包括：相当数量的软件开发人员没有掌握正确的软件开发方法学，对于软件开发与维护存在许多模糊、错误认识。对于软件开发的典型错误认识有：

- 只要初步了解总目标，就可以编写程序，细节在编程中解决；
- 软件更改容易，因此用户需求可以在程序设计过程中逐步补充；
- 程序是软件的主要部分，文档是可有可无的；
- 软件开发依赖于开发人员的创造力与想象力，程序设计是一门艺术，因此软件设计无章可循；
- 一旦软件能够运行，维护是不需要的，或者说是极少的。

……

这些错误认识与方法的形成可以归于软件人员没有掌握恰当的工程化方法，忽视问题定义与分析，急于求成，闭门造车。一个软件错误修改的代价随该错误发生与修改滞后的时间关系极大。一个早期发生的错误修改越晚，代价越大（如图 1.1 所示）。事实上，许多软件工程项目失败的主要原因就是对问题定义、用户要求没有认真的分析，在没有弄清用户到底要求什么时便开始编程。这如同一个大厦的建立没有地基，最终必然垮台。

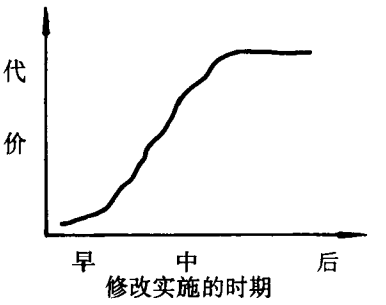

图 1.1　错误修改的代价

软件管理人员的缺陷主要指软件管理技术落后，软件质量管理缺乏，往往也难以遵循规范的标准，管理没有依据。在软件配置管理中，文档一致性、完整性差，项目管理中费用估计不足，开发进度无序，无法保证软件技术方案的实现。

60 年代出现的软件危机使人们认识到软件开发组织与方法必须参照传统工程（如机械工程，建筑工程等）的方法学。总结计算机软件系统研究与开发的经验教训，探索与实施既遵循一般工程组织实施原理，又符合软件开发自身特点的方法与技术，建立软件工程规范。因此掌握正确的软件工程方法学是解决问题的首要任务。一般工程组织的方法是将任务分割为子任

务,确定计划目标,分期实施。生产过程监控、分阶段测试是工业产品质量控制的主要措施。将一般工程方法学的这些基本原理移植到软件开发与维护中来,使软件从开始提出到最后丢弃整个过程划分为计划、分析、设计、实现、测试、维护各个子阶段,规定各阶段的任务及完成标志(里程碑),审查标准。将软件开发技术与管理技术相结合,使得软件作为工程产品处理,期望能够解决软件危机的各种问题,这是人们对于软件认识的一次飞跃。

软件工程又不同于一般传统工程。软件产品与开发过程具有不可见性,软件开发的管理对象是问题求解的方法、流程、算法、概念等。因此文档是软件工程组织有形化的主要途径。软件工程主要通过文档实现对人员组织,项目进度,质量控制,系统配置的管理。使软件开发能够实现低成本,高质量,按期交付。

"工欲善其事,必先利其器"。要提高软件开发效率,必须开发先进的软件工具。软件开发本身是高度智力密集型劳动,它所使用的工具应该能够"放大"人的智能。在软件开发各阶段,诸如文档整理,版本维护等需要许多繁杂重复的劳动,这些工作应该自动化。因此智能化,自动化(至少是半自动化)是人们对软件工具的期望。将各种软件工具有机的集成起来构成一个相互配合、协作的,能够支持软件开发全过程的系统,这就是软件开发环境。引入人工智能技术,数据库技术,将特定的软件工程方法学与自动化软件工具系统相结合,使得软件开发与维护中各种手工的繁杂重复性劳动能够智能化、自动化,这就是计算机辅助软件工程(CASE,Computer-Aided Software Engineering)。

软件开发与维护方法学,软件工具与环境,软件管理与质量控制等等,对这些问题的研究综合在一起形成了软件工程学。IEEE 将软件工程术语定义为:以优质、高效、低成本为目标,研究开发、运行和维护软件以及使之退役的系统方法。

因此,软件工程是指导计算机软件开发与维护的工程科学。它借鉴传统工程的原则、方法,应用计算机科学、数学、管理学的基本原理、技术来开发与维护软件。以期实现高质量、高效率、低成本的生产软件产品。其中计算机科学、数学用于构造软件系统模型与算法,工程科学用于制定规范、标准,管理科学用于计划、资源、成本、质量控制。

软件工程的目标是追求软件产品的正确性、可用性和软件生产的效率。其工程活动包括需求分析、设计、实现、测试确认、维护支持,通过这些活动实现其目标的方法学问题包括:开发模型、设计方法、支持技术、管理过程。

1.2 软件开发模型

1.2.1 软件生命周期

软件生命周期是指一个软件系统从目标提出到最后丢弃的整个过程。软件工程基本原理强调软件生命周期的阶段性,其基本思想是各阶段任务相对独立,具有明确的完成标志。阶段的划分使得人员分工职责清楚,项目进度控制和软件质量得到确认。原则上,前一阶段任务的完成是后一阶段工作的前提和基础;而后一阶段的任务则是对于前一阶段问题求解方法的具体化。整个软件生命周期可以划分为三个阶段,即软件定义、系统实现和运行维护。各阶段的划分如图 1.2 所示。

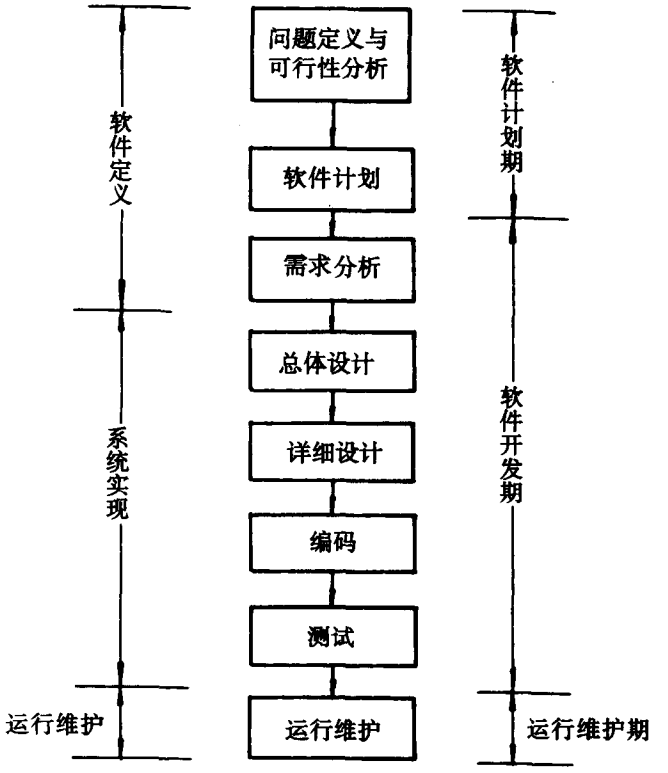

图 1.2　软件生命周期的阶段划分

1. 软件定义

软件定义阶段的核心是“要求解决什么问题?”围绕这一核心应该完成的任务是:确定软件开发总体目标,进行可行性研究,制定软件开发计划,准确定义系统功能、性能结构,为软件设计提供依据。因此软件定义进一步可以分为问题定义与可行性研究,软件计划,需求分析三个子阶段。

(1) 问题定义与可行性研究。根据用户或者市场需求,提出软件项目目标和规模。即确定“要解决什么问题?” 显然一个公司的软件项目应该由其高层决策人员提出,然后由信息处理部门经理或者系统分析员定义问题的性质,目标和系统规模并以书面报告提交。问题求解目标一经提出,分析员必须对它进行可行性研究。可行性研究必须回答的是:“问题是否有解?是否值得求解。”因此可行性研究报告应该构造系统求解的高层逻辑模型,该模型更准确、具体的描述系统规模与目标,同时对系统开发成本与将来可能获得的效益进行准确估计。对建议实施的软件项目进行成本效益分析是可行性研究的主要任务之一。

(2) 软件计划。如果建议中的软件项目可行性研究的答案是肯定的,则要求制定软件开发计划。软件开发计划的主要内容是规定系统目标,软件开发的资源配置(资金,人力,硬、软件系统支持),项目组织,进度控制及各阶段确认标准。软件计划的完成标志是软件项目计划书。

(3) 需求分析。需求分析的任务是完整准确地定义系统必须“做什么?”并用开发人员与用户均能准确理解的语言表达出来。需求分析的问题通常是:一方面,用户知道应该解决什么

问题,但不能够完整准确地用信息处理的语言表达出来,不知道计算机系统可以干什么。另一方面,软件开发人员熟悉如何用软件技术实现人们的要求,但并不知道特定用户的具体要求,不熟悉用户业务细节。因此需求分析通常由系统分析员主持,软件技术人员与用户共同进行,其完成标志是软件需求规格说明书 SRS(Software Requirement Specifications)。

需求分析是软件开发的基础性工作,必须高度重视,谨慎实施。工程实践中许多项目的失败根源往往就是没有进行认真的需求分析。需求分析文档 SRS 描述了经过用户确认的系统逻辑模型,它既是软件设计实现的依据,同时也是项目最后验收交付的依据。

从软件项目的组织角度看软件定义的第 1,2 子阶段完成的是软件项目的前期准备工作,这部分工作应该用尽可能小的开销在较短时间内完成。典型的软件项目可行性分析与计划通常由一个分析员或者项目论证小组在不超过半个月的时间内完成。当然对于重大投人和大规模开发的项目要求进行慎重的论证,可能需要更多的时间和开销。需求分析是软件项目进入实际开发的第一个阶段,这里要求进行大量的调查、理解、分析和文档工作,典型情况下需求分析的工作量约占软件开发期的 15%～20%。

2. 系统实现

系统实现阶段核心是"如何解决问题?"围绕这一核心应该完成的任务是:设计和实现 SRS 定义的软件系统并交付给用户。贯穿于软件实现整个过程的基本思想是采用对问题进行分解的方法,自上而下,逐步求精。因此软件系统实现通常分为总体设计,详细设计,编码,测试四个子阶段。

(1) 总体设计(概要设计)。总体设计的任务是确定目标系统的解决方案,即"总体上,应该是如何解决这个问题"。其工作包括系统功能设计和系统结构设计。

系统功能设计的任务是确定系统的外部规格与内部规格。所谓外部规格包括:系统运行环境,用户可见功能、性能一览表,系统输人/输出数据格式。内部规格是指各主要处理(包括异常处理)的基本策略,系统文档种类与规格,系统测试总体方案。系统结构设计的任务是确定系统模块结构 ,确立各模块功能划分和接口规范,调用关系。确定主要模块算法和主要数据结构。

总体设计通常由系统分析员或者高级程序员担任,其完成标志是总体设计说明书。总体设计说明书中系统功能设计可以采用表格形式给出,而系统结构设计通常由软件结构图或者高层 IPO 图给出。

(2) 详细设计。详细设计的任务是使用某种形式化语言描述总体设计在规定环境中的实现,即"应该如何具体地实现这个系统"。其工作是根据总体设计说明书给出各模块内部的数据结构和算法描述。详细设计由高级程序员和程序员担任,按照系统结构将各模块分解到人。详细设计的完成标志是用图形或者伪码描述的模块设计说明书。

总体设计与详细设计合在一起又称为系统设计,是软件开发期的主要阶段,通常占软件开发全部工作的 25%～40%。

(3) 编码实现。编码的任务是根据模块设计说明书,用指定的程序设计语言写出正确、易理解、易维护的程序模块并调试通过。在保证完成质量的总体设计与详细设计基础上,编码通常所做的只是将详细设计的图形描述或者伪码翻译为指定程序语言代码。

系统编码的完成标志是可运行代码和完整的模块内部文档(源程序清单)。编码是开发期中相对简单的阶段,通常占软件开发全部工作的 15%～20%。

(4) 软件测试。软件测试的任务是通过各种类型的测试使软件达到预期的要求。测试工作按照测试对象和目的可以分为单元测试、集成测试和验收测试。

单元测试又称模块测试,依据通常是模块设计说明书。其任务是测试单个模块实现的功能、性能。因此这部分工作常常在编码阶段与模块调试一起进行。但是系统重要的,关键的模块应该独立进行单元测试。

测试阶段的主要任务是系统集成测试和验收测试。集成测试根据系统总体设计将经过单元测试验收的模块按照某种指定策略装配起来,在装配过程中测试各模块相互的接口和调用关系,考察各模块集成以后是否能够准确实现系统功能设计规定的各项功能和性能。验收测试则是依据 SRS 由用户对系统进行验收。

测试是保证软件质量的重要手段,通常占软件开发全部工作的 40% 以上,复杂系统测试可能占到软件开发工作的 2/3。大型软件的测试通常由独立的部门和人员进行,通过测试结果的分析,要求建立系统可靠性模型,对系统可以达到的各项功能、性能指标进行量化确认。因此测试文档包括测试计划,测试方案(测试用例与预期结果),测试报告(测试结果分析与可靠性模型)。测试的完成标志是系统测试报告。

3. 运行维护

运行维护的任务是通过各种维护活动使系统能够持久地满足用户需要。

软件维护的具体活动包括纠错性维护,适应性维护,功能性维护和预防性维护。所谓纠错性维护就是改正在软件运行过程中暴露出来的系统遗留的各种错误,这种维护活动在系统交付初期比较频繁,但当系统进入稳定运行期后应该很少发生(如图 1.3 所示)。适应性维护是指当系统运行环境发生变化以后,为适应这种改变必须对软件进行的修改。功能性维护是指在软件过程中为满足用户需求的变化与扩充对软件所做的修改。预防性维护则是指为改善软件将来的可维护性所做的准备工作。软件运行稳定以后,维护的主要活动应该是适应性和功能性维护。

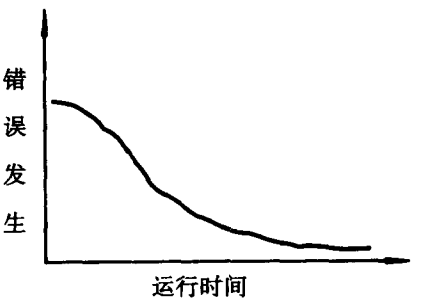

图 1.3　软件投入运行后的错误

必须注意的是,每次维护活动的发生都是对软件的修改,逻辑上是一次软件重新定义和设计,很有可能带来新的错误。因此每一项维护活动都包括提出维护要求(问题报告),维护计划制定与批准,维护方案分析,修改设计,修改程序,回归测试,复查验收等一个压缩的软件开发全过程。

软件运行的日常状态和问题应该详细记载,这就是软件运行日志。当每次维护活动发生以后必须提交准确记录本次维护各项活动的维护报告并修改用户使用、维护手册的相应部分,同时还应该保留系统修改以前的版本。

系统维护的工作量十分巨大,通常可能占到计算机系统软件总成本的 70% 以上。但是由于运行维护持续的时间很长,因此一个正常运行的软件系统在单位时间内的维护费用应该是合理的。如果一个软件的维护要求巨大的支出,那么就应该考虑软件系统的更新,也就是该丢弃这个软件了。

1.2.2 软件开发的瀑布型模型

严格按照软件生命周期的阶段划分，顺序执行各阶段构成软件开发的瀑布型模型，如图1.4所示。瀑布模型的特点是：

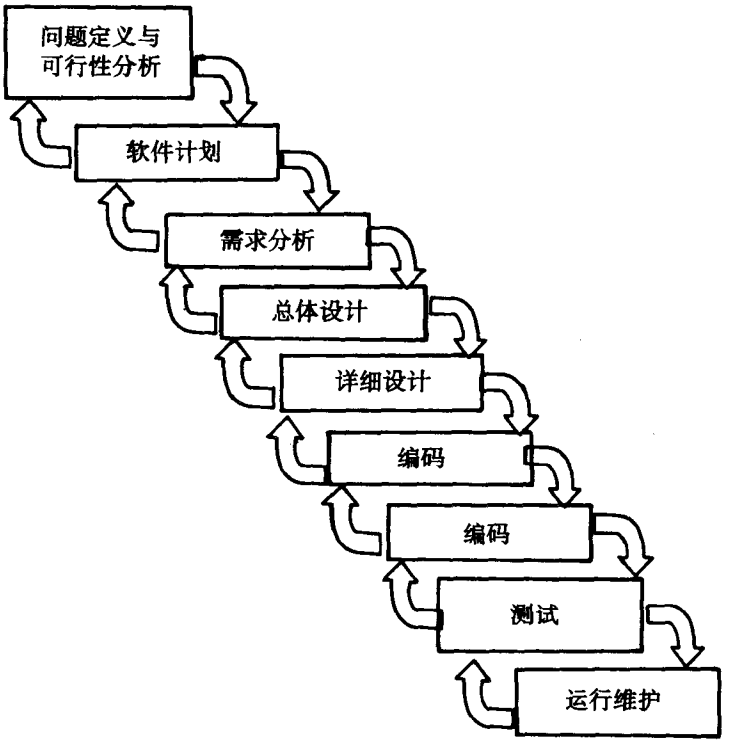

图1.4 软件开发的瀑布型模型

1．阶段间具有顺序性和依赖性

顺序性要求每个阶段工作开始的前提是其上一阶段工作结束。因此前一阶段输出的文档就是其后一阶段的输入文档。依赖性是指各阶段工作正确性依赖与上一阶段工作的正确性。因此当某一阶段的工作出现问题时不仅要考察本阶段的工作，还必须追溯前面阶段的工作。

2．推迟实现的观点

软件开发失败的许多实例都是软件人员急于系统编码实现。编码开始的越早，项目完成的时间很可能越长。这是因为过早进入编码往往意味着大量的返工。因此瀑布型模型明确要求在项目前期，即需求分析和总体设计阶段只考虑系统的逻辑模型，不涉及系统物理实现。即详细设计，其重点也是数据结构与算法设计，要求仅仅用图形或者伪码描述。因此，直到设计结束，软件开发人员考虑的主要是系统逻辑实现模型。将系统逻辑设计与物理实现严格分开，尽可能的推迟物理实现，这是瀑布型模型的一个重要指导思想。

3．质量保证的观点

为保证软件开发质量，瀑布型模型在生命周期的各阶段强调：

第一，制作规定的文档是各阶段完成的里程碑，没有交出合格的文档也就没有完成该阶段