装备软件质量和可靠性管理

ZHUANGBEI RUANJIAN ZHILIANG HE KEKAOXING GUANLI

阮 镰 陆民燕 韩峰岩 编著

国防工业出版社
National Defense Industry Press

装备软件质量和可靠性管理

阮镰　陆民燕　韩峰岩　编著

国防工业出版社
·北京·

图书在版编目(CIP)数据

装备软件质量和可靠性管理/阮镰等编著. 北京：国防工业出版社，2006.1

ISBN 7-118-04235-8

Ⅰ.装... Ⅱ.阮... Ⅲ.软件质量 质量管理 Ⅳ.TP311.5

中国版本图书馆 CIP 数据核字(2005)第 133641 号

※

国防工业出版社出版发行

(北京市海淀区紫竹院南路 23 号 邮政编码 100044)

北京奥鑫印刷厂印刷

新华书店经售

*

开本 787×1092 1/16 印张 17¾ 字数 406 千字

2006 年 1 月第 1 版第 1 次印刷 印数 1—4000 册 定价 40.00 元

(本书如有印装错误，我社负责调换)

国防书店：(010)68428422 发行邮购：(010)68414474

发行传真：(010)68411535 发行业务：(010)68472764

前　言

现代高新技术在航空装备研制中的应用日趋广泛，其标志之一是数字化的设备大量应用于航空装备。因而计算机软件在航空装备中得到大量的应用，不论在其规模上，还是在其重要性上均呈急剧上升的趋势。

表 1 展示了美国第二、三、四代歼击机上，航电系统中由硬件和软件承担其功能的百分比，由表 1 知：美国的歼击机每更新一代，其由软件实现的功能翻一番。

表 1　软件在美国航电系统中的应用

第 X 代飞机	型　号	航电系统功能	
		硬件实现	软件实现
第二代	F-111	80%	20%
第三代	F-16	60%	40%
第四代	F-22	20%	80%

在国内，航空装备的状况也相类似，我国新研的军用飞机中，其飞控系统、火控系统、发动机燃油调节系统，甚至弹射救生系统等均采用了由软件实现其控制功能，逐步取代了原有的机械或光学设备。在某些机种上机载软件已超过百万行源代码。

毫无疑问，软件在飞行器中起着越来越重要的作用。但不幸的是，软件的质量和可靠性却远不如人意。特别是与硬件可靠性相比，软件的可靠性一般要比硬件低一个数量级。表 2 是美国一个权威机构在 2000 年 3 月发布的，关于美国软件开发质量的统计表。

表 2　美国软件开发质量统计表

- 软件项目中途终止的占 25%；
- 软件产品在交付时通常在产品中还残留 15%的缺陷；
- 软件公司花在软件返工(修改)上的资源为 30%～44%；
- 软件失效往往比硬件失效高一个数量级

与美国的软件开发水平相比，我国的航空装备软件开发水平还相差很远，因而软件的质量和可靠性水平更令人担忧。在研的航空装备上装备着大量的计算机软件，软件故障已成为引发飞行器故障的主要原因。不少事实证明，软件故障是导致航空型号失败的重要因素。

要保证航空装备的质量，确保装备的战斗力，提高软件的质量与可靠性是最关键的一个途径，这已成为航空界的共识。为此，本书便应运而生了。

本书试图从软件开发全过程的角度探索软件质量与可靠性的管理框架及其相应的方法和程序。我们给出了一个现代软件质量与可靠性管理的三维模型：时间维——对软件生存期的全过程控制；空间维——对软件开发质量的全方位管理；组织维——为软件开发构建三位一体的层次管理体系；从而从时间、空间和组织等诸方面对软件质量进行控制。

本书还对软件可靠性的概念，方法和管理进行了专门的论述。

因此，本书不仅适用于从事航空软件开发和管理的人员，亦适用于其余行业从事装备软件开发和管理的人员。

与一般只面向软件开发者的书籍不同，本书同时还面向装备软件的使用者，对使用方在装备软件开发过程中的监控进行了全面的阐述。

本书共分六章：

第1章　关于软件质量和可靠性基本概念的概述。

第2章　阐述时间维中对软件生存期全过程的控制，我们对软件的每个生存期阶段都给出了各阶段的进入条件、主要任务、基本要求、主要质量控制手段、主要质量度量、方法和工具以及阶段产品和阶段完成的标志。

第3章　阐明空间维中对软件开发质量的全方位管理，包括软件的分级管理、文档管理、需求管理、评审管理、配置管理、测试管理、软件FRACAS管理和分承包单位管理。

第4章　叙述组织维中，对软件开发者个人、软件开发小组和软件开发单位构建三位一体的层次管理体系，并给出了该管理体系中的管理目标、量化目标、开发过程质量评价准则和软件产品质量评价准则。

第5章　论述软件可靠性工程的内涵和方法。鉴于软件可靠性工程方面的书籍甚少，为给读者一个软件可靠性工程的概貌，在本章中我们阐述了包括如何确定软件可靠性参数与指标、如何进行软件可靠性设计和验证，如何进行软件可靠性评价和软件可靠性管理等内容。

第6章　是关于装备使用方在软件开发过程中的作用、监控方法和手段；以及使用方型号办，军事代表在软件开发中的作用的阐述。

为方便读者使用，本书在附录中还分别给出了按国军标438A编制软件需求规格说明、设计文档、测试文档等详尽的编写指南。

本书第1章～第4章由阮镰教授撰写，第5章由陆民燕教授执笔，第6章由韩峰岩高工撰写，全书由阮镰主编。

本书在编著过程中得到了北京航空航天大学计算机学院的金惠华教授（博导）、航天科工集团706所的王纬研究员和兵器工业集团系统所李良巧研究员（博导）的热情帮助，他们在百忙中仔细地审阅了全书，并提出了许多宝贵的意见。

本书是由空军航空技术装备可靠性办公室支持出版的，没有他们的大力支持，本书的写作和出版是不可能的。

承担部分编写工作及提供材料的还有王纬、张虹、曾福萍、鲍晓红、吴玉美、杨晓豫等人。

对于所有帮助和支持本书写作与出版的同志，我们表示深深的谢意！

2005.10　北京

目　录

第 1 章　软件质量与可靠性管理概述

1.1　软件质量和可靠性管理的发展历程

众所周知，在软件的开发初期是没有软件质量和可靠性管理的，直到 20 世纪 60 年代末，在世界范围内爆发了“软件危机”后，人们才认识到，只靠编程语言、编程方法来开发软件，尤其是大规模的复杂软件是不行的。必须有一套工程化、规范化的方法来指导软件的开发和管理。于是，在 70 年代初，就有了软件生存期的第一个模型——瀑布模型(WFL)和软件工程(SE)。在 70 年代中期开发出了第三代编程语言(3GL)。在 80 年代初，则推出了全面质量管理(TQM)。80 年代中后期有快速原形技术(PROTO)和计算机辅助软件工程工具(CASE)。在 90 年代，软件质量管理上了一个新台阶，即从软件产品的质量管理发展为对软件开发过程的过程管理。这一标志是 1991 年美国推出的 CMM(即软件能力成熟度模型)和在 90 年代中期推行的 SPI(软件过程改进)，包括 PSP(个体软件过程)和 90 年代末的 TSP(小组软件过程)等。

软件可靠性则是在 20 世纪 70 年代中期提出，到 80 年代中期得到发展，大约在 1988 年，技术人员开始使用软件可靠性方法。AT&T Bell 实验室围绕软件可靠性工程(SRE)的概念进行了一系列的工作，软件可靠性的工作已超出了软件可靠性的建模和测量，扩展到了软件可靠性测试和软件可靠性管理。1990 年 IEEE 的软件可靠性工程分委会成立，而到了 90 年代中期才发展出了软件可靠性工程，其中当然包括了软件可靠性管理。

图 1.1 清晰地显示了从 20 世纪 70 年代直到 2000 年，软件质量和可靠性管理以及软

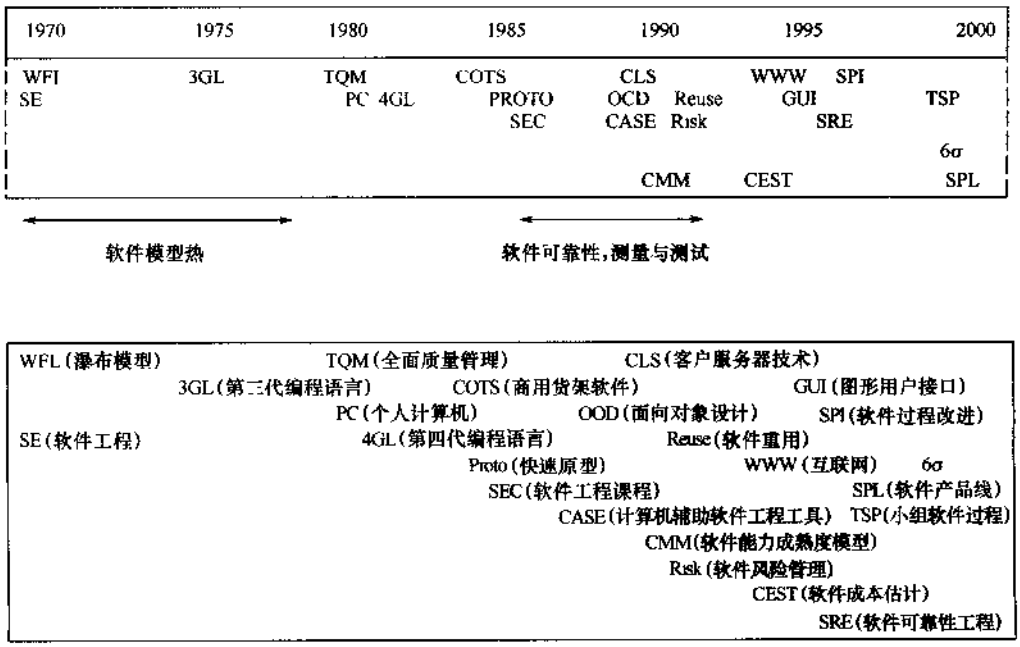

图 1.1　软件质量和可靠性管理的发展历程[1]

件工程的发展历程。

1.2 软件质量的基本概念

1.2.1 软件质量、质量管理与质量控制

(1)软件质量。软件质量是软件产品满足用户使用要求的程度。

(2)软件质量管理。软件质量管理是在软件质量方面指挥和控制组织的协调的活动。

(3)软件质量控制。软件质量控制是对开发可用软件产品的过程的测量与监控。

如上所定义,软件质量是软件产品的一组固有特性满足用户使用要求的程度。为了使软件产品质量满足用户使用要求,必须实施软件质量管理。我们从软件质量管理的角度讨论过程控制,实际上是讨论软件生存期过程特别是软件开发过程的质量控制,只要这些过程在质量方面得到恰当的控制,所开发的软件产品的质量就会使用户满意。

需要说明的是,本书并不涉及质量管理的所有方面。因为,质量管理是在质量方面指挥和控制组织的协调的活动。在质量管理方面的指挥和控制活动,通常包括制定质量方针和质量目标以及质量策划、质量控制、质量保证和质量改进。而本书主要涉及质量控制。质量控制是质量管理的一个重要部分,致力于满足质量要求。

软件质量控制的核心是过程控制。

1.2.2 软件的质量特性

软件的质量特性是一组描述和评价软件产品质量的属性。

据 ISO/IEC 9126-91 软件质量可定义为 6 个特性和 27 个子特性。

软件的 6 个质量特性是:

(1)功能性(functionality):当软件在指定条件下使用时,软件产品满足规定需求和隐含需求的功能的属性。

(2)可靠性(reliability):在规定的条件下、规定的时间区间内,软件实现其规定功能的能力。

(3)易用性(usability):在指定条件下使用时,与用户使用软件所需努力程度有关的属性。

(4)效率(efficiency):在规定条件下,软件产品可提供的性能水平与其所用资源的数量相关的属性。

(5)可维护性(maintainability):软件产品可被修改、维护的能力。

(6)可移植性(portability):软件产品从一种环境转移到另一种环境的能力。

以上 6 个特性及其派生的 27 个子特性的关系可参见图 1.2。

据 JohnMusa 在《软件可靠性工程》中的论点,与 IEEE 于 2004 年 AR&MS Tutorial Notes《Software Reliability and Maintainability》中的论述:均认为在软件质量属性中,"可靠性是最重要的软件质量属性"。

关于软件可靠性的基本概念可参见第 5 章。

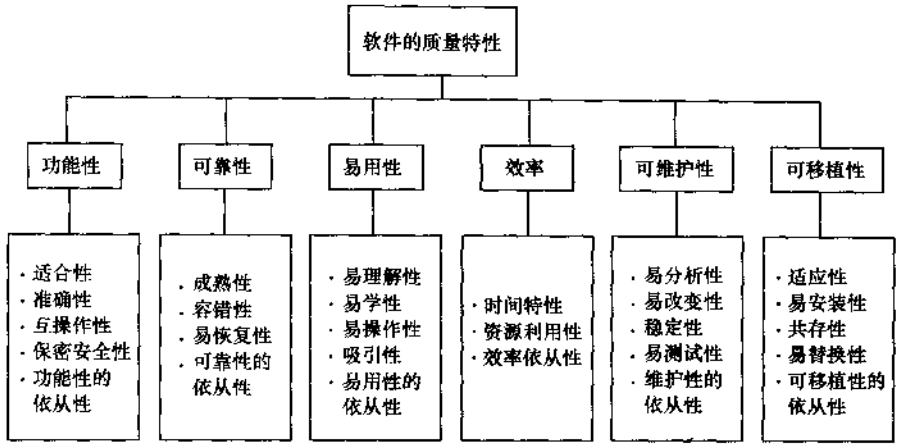

图 1.2　软件的质量特性及其子特性

1.2.3　软件的质量模型

对于软件的质量特性，必须予以量化与测量。因为对任何事物，如果你不能测量它，你也就不能控制它。("If you can't measure it, you cann't control it!")对软件的上述质量特性建立量化的模型是为了更好地控制软件质量。

1. 功能性模型

软件的功能性是软件产品的功能满足用户需求的程度。软件功能性模型 F：

$$F=\frac{\text{软件产品能提供的功能数 } n}{\text{用户在需求规格说明中要求的功能数 } N}$$

这里的功能是指广义的功能，包括性能(功能实现的准确程度)、互操作性、完整性等特性在内。

2. 可靠性模型

软件可靠性是与规定的时间区间内，软件无失效运行的状况相关的特性。软件可靠性模型 R 一般可为：

$$R=\frac{\text{规定的软件运行时间区间 } T_R\text{(小时)}}{\text{在规定时间区间内软件的失效数(个)}}$$

3. 易用性模型

软件易用性是与使用软件的努力程度(如培训、操作难易程度)相关的特性，可给出易用性模型 U：

$$U=\frac{\text{为使用软件的学习和培训工作量(人—时)}}{\text{软件开发工作量(人—时)}}$$

U 越小，表示软件越容易使用。

4. 效率模型

软件效率是与计算机软件资源利用率(如存储空间、处理时间、通信时间等)相关的特性，效率模型 E 为：

$$E=\frac{\text{软件实际利用的资源}}{\text{分配给软件的可利用资源}}$$

E 太小，就表示实际利用的资源太小，存在资源的浪费；但 E 也不能过大，一般要求

$E \geqslant 0.8$，即对存储空间、处理时间等要求至少有20%的余量。

5. 可维护性模型

软件的可维护性是当软件失效后，与软件故障定位、排除的难易程度等有关的特性。可维护性模型 M 为：

$$M=\frac{\text{需要维护与修正软件所有故障花费的总时间 } T_M\text{(小时)}}{\text{需要维护与修正的软件故障总数(个)}}$$

M 越小，表示软件的可维护性越好。

6. 可移植性模型

软件的可移植性是软件在另一个环境(硬件配置与/或软件系统环境)使用时，与其转换工作的努力相关的特性。可移植性模型 P 为：

$$P=\frac{\text{将软件移植到另一环境中所花的工作量(人一时)}}{\text{软件开发工作量(人一时)}}$$

P 越小，表示软件的可移植性越好。

1.2.4 软件可靠性工程模型

1.2.4.1 软件可靠性工程 Musa 模型

图1.3是John Musa给出的软件可靠性工程的过程框架，姑且称之为软件可靠性工程的Musa模型。该模型将软件可靠性工程过程定义为：

- 定义“必要的”可靠性；
- 开发运行剖面并准备测试；
- 执行测试及失效数据处理；
- 得到软件的可靠性。

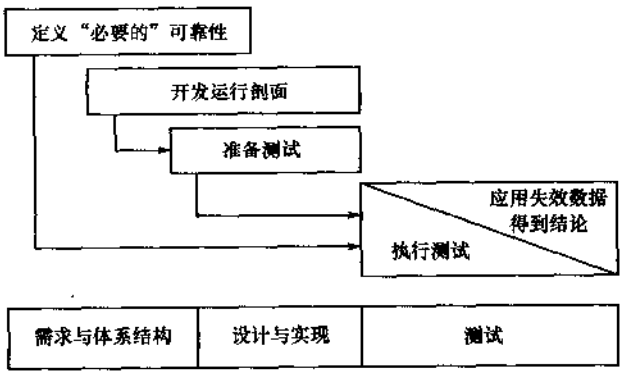

图1.3 软件可靠性工程过程框架(Musa)[2]

我们认为此模型的不足之处在于：

(1)软件可靠性工程遍历的阶段只有需求、设计、实现和测试阶段，软件可靠性工程应当覆盖软件生存期全过程，软件可靠性的增长一直要延续到使用维护阶段。

(2)缺少软件可靠性分析与设计，软件可靠性是设计出来的，在定义“必要的”可靠性参数和指标后，接下来应进行软件可靠性分析和设计。

(3)仅仅定义“必要的”可靠性是不够的，还应进行软件可靠性估计，包括软件可靠性

早期预计和增长预计。

为此，我们给出了软件可靠性工程框架。

1.2.4.2 软件可靠性工程 Ruan 模型

图 1.4 展示的软件可靠性工程模型是作者对图 1.3 所示的 Musa 模型的改进：

(1)它包括了软件的整个生存周期，这意味着软件可靠性工程活动必须覆盖软件生存周期，而不是到测试阶段就结束。

(2)它包含了软件可靠性分析和软件可靠性设计的重要内容，亦即软件可靠性是设计出来的。

(3)它不仅包括了软件可靠性参数与指标的确定(即"定义必要的可靠性")，而且还涉及了对这些指标的预计，这对可靠性设计方案的权衡、选择十分重要；还包括在测试阶段和使用、维护阶段进行的软件可靠性预计和估计。

对软件生存期中各项可靠性工程活动详细的阐述参见本书第 5 章。

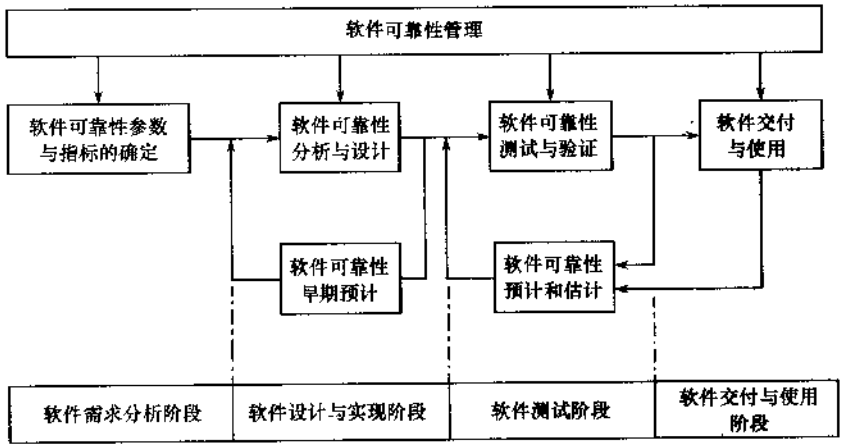

图 1.4 软件可靠性工程框架(Ruan)

1.2.5 软件质量的 Pareto 原理

Pareto 是意大利的经济学家，他在分析社会财富分配状况时得出了"20%的富人，掌握了 80%的社会财富，其他 80%的人只占有 20%的社会财富"。这一原理随后被广泛地应用于其他领域，如系统工程中的"八·二定律"，硬件研制中对硬件系统关键件的分析等。

Pareto 原理在软件开发中也被证明是正确的，Barry Boehm 给出了软件质量和软件现象所遵循的 Pareto 原理[3]：

(1)20%的软件模块包含了软件中 80%的缺陷；

(2)20%的软件缺陷，消耗了 80%的更改维护费用；

(3)20%的软件改进，花费 80%的适应性维护费用；

(4)20%的模块占用了 80%的执行时间；

(5)20%的模块消耗了 80%的资源(人力、经费等)；

(6)20%的软件工具占用了 80%的全部工具使用时间。

本书作者再补充二条重要的软件质量的 Pareto 原理：

(7)20%的软件缺陷，造成了 80%的软件故障；

(8)20%的软件开发和管理人员(骨干)，决定了 80%的软件开发质量。

根据 Pareto 原理，质量管理之父 Juran 提出了在质量管理中存在着“少数的重要事情(Vital few)”和“多数的琐碎事情(trivial many)”的观念。这就给软件质量管理指明了方向：在有限的资源条件(有限的人员、时间、经费)下，抓软件质量要抓“少数的重要事情”，要抓 20%的关键模块，对这些关键模块的开发过程必须从文档、评审、配置管理、测试等方面严加控制，就可有效地提高软件质量。

1.3 软件质量与软件工程化

如何保证软件产品质量，一直是软件工程界十分关注，致力解决的问题。20 世纪 60 年代末发生的世界软件危机的主要表现之一就是软件质量差。当时软件界共同探讨解决软件危机的方法，得出的唯一结论就是吸取硬件的工程经验用于软件开发，即实施软件工程。30 多年来国际软件工程界一直在探索和发展软件工程的实施方法，如今软件工程已经成为一个专门学科。虽然“软件危机”问题尚未完全解决，但实践已经表明，实施软件工程的确是保证软件质量，解决软件危机的唯一有效方法。

实施软件工程，确保软件质量的核心就是用软件工程方法组织软件开发。

众所周知，产品质量主要取决于产品研制过程的质量。软件产品也是如此，软件质量主要决定于软件的开发过程。由于软件产品是“通过承载媒体表达的信息所组成的一种知识产物”，软件产品一旦形成，在没有人为改动的情况下，就有“千篇一律”和“一成不变”的特点。因此，软件产品的质量主要由软件开发过程决定。

用软件工程化方法组织软件开发包含两方面的内容：

(1)用软件工程方法开发软件，即软件的工程化开发；

(2)用软件工程方法管理软件开发，即软件的工程化管理。

要确保航空装备软件的质量，必须一方面重视软件的工程化开发，另一方面重视软件的工程化管理，两者不可或缺。这也是人们通常所说的软件的质量是设计(开发)出来的，也是管理出来的。

1.3.1 软件的工程化开发

软件的工程化开发，就当前的飞行器软件而言，主要与软件开发的方法学有关。

当前的软件开发方法学主要有：

1. 结构化方法

结构化方法包括结构化的分析、结构化的设计、结构化的编程和结构化的测试。

结构化方法认为软件系统是以一定的结构形式存在，由若干子系统构成。软件系统可以一定的准则，自顶向下进行层次分解，直至分解到低层次的模块。

在飞行器中应用的大量嵌入式软件，现在大多仍采用结构化的开发方法。

2. 面向对象的方法

面向对象的方法是以对象为中心构造模型、组织软件系统。这种方法认为客观世界

由对象组成，不同对象间的相互作用和联系构成了不同的系统。应用计算机解决问题的方法空间应当与客观世界的问题空间相一致。

面向对象方法中的对象是由数据及其上的操作组成的封装体；对象是类的实例；面类则是具有相同属性和服务的对象的集合。

3. 净室(Clean room)方法

净室方法是在结构化分析和设计方法的基础上，增加了需求分析和设计的形式化方法。这种方法认为软件程序设计组应努力开发出在进入测试之前就几乎无错的系统。

4. 形式化方法

形式化方法以严格的数学证明为基础，要求软件需求规格说明要用形式化的语言描述，以保证其正确无误，然后经过一系列变换直到产生出可执行程序。

由于这种方法需要较多的数学验证，目前尚缺少可支持的工具，因此在航空装备的工程研制中尚未应用。

在航空装备的软件开发中，选择何种开发方法学取决于软件项目的特点、能得到的支持环境和技术支持，以及开发人员的技术水平和经验等因素。

鉴于上述软件开发方法学在一般软件工程书籍中多有阐明，本书不再赘述。

1.3.2 软件的工程化管理

软件的工程化管理极其重要。总结国内、外飞行器软件开发的经验和教训，可以认为管理仍是决定软件质量的关键所在。

软件工程化管理是指对于一个软件工程项目为了确定和满足需求所必须进行的一系列组织、计划、协调和监督等工作。多年来，经过大量调查研究发现：管理仍然是开发软件项目成败的关键。

早在 20 世纪 70 年代中期，美国国防部就组织力量研究软件项目失败的原因，发现在失败的软件项目中，70%是由于管理不善所造成的，因而认为管理影响全局，并掀起了研究软件管理技术的热潮。20 年后，根据美国三份经典研究报告，这一状况并未得到转变：软件开发仍然很难预测，大约只有 10%的项目能够在预定的费用和进度下交付符合需求的软件；管理仍然是软件项目成败的主要因素；并指出开发过程中的返工是软件过程不成熟的标志。

软件工程化管理具有以下几个特征：

- 没有适当的管理，软件开发不可能完成好，也就谈不上软件工程化；
- 软件工程项目越大、越复杂，管理工作量占整个软件研制工作量的比例也越大；
- 管理的基本目标是以最小代价达到对工程项目预定的要求，基本任务是保证恰当地确定软件需求和圆满地实现需求。

软件工程化管理的关键是：

- 对软件开发过程的全过程控制；
- 对软件质量的全方位管理；
- 建立多层次的软件开发、管理体系。

1.4 软件质量管理的基本框架——现代软件开发与管理的三维模型

本节提出的软件质量管理的基本框架体现了现代软件工程化管理的理念，为软件开发与管理构建一个全方位、全过程、多层次的三维框架，这三维是：

(1)时间维：对软件生存期的全过程进行控制；

(2)空间维：对软件质量有关的关键因素实施全方位管理；

(3)组织维：构建从软件开发个人、软件开发小组到软件开发单位的多层次的管理模式。

图 1.5 展示了三维管理体系的框架。

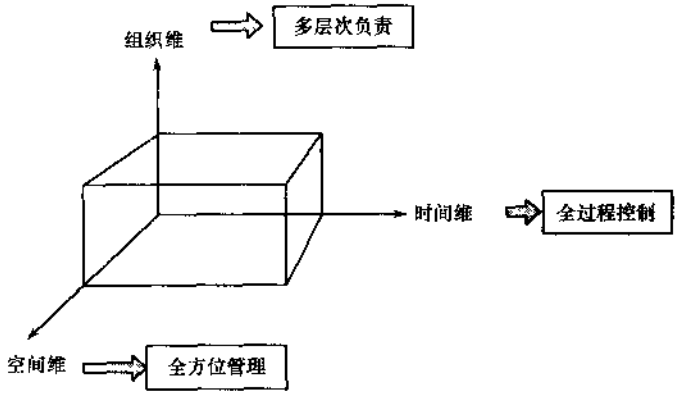

图 1.5 现代软件工程化管理的三维框架

1.4.1 时间维——对软件生存期的全过程控制

1.4.1.1 过程的定义

(1)过程(process)：通过人员，设备、器材和规程(procedures)的交互作用，以期提供一个规定的服务，或生产一个规定的产品(如图 1.6 所示)。

每个过程是一组(或多组)被定义的，应在某个给定时间内完成的活动(activities)或任务(tasks)。这些活动的特点是能被可测量的、可控的输入和输出所描述。

(2)软件开发过程：将用户要求/需求转化为软件产品的过程。

为了确保有效的质量控制，管理者应当将开发过程分解为子过程，并满足以下的质量准则：

①降低开发过程的复杂性以确保可见性(visibility)和可控性；

②标识出对质量和项目成功至关重要的任务；

③确定并按时间顺序指定各控制点，在这些点上能验证过程的正确趋势和结果的正确输出；

④在项目特定的组织结构中对所有质量功能提供清晰的标识和分配；

⑤使用已证实的和正确推导以及解析的质量度量；