刘宗田 潘秋菱 李 钢 李心科 等著

软件质量评价与保障

RUANJIAN ZHILIANG PINGJIA YU BAOZHANG

上海大学出版社

图书在版编目(CIP)数据

软件质量评价与保障/刘宗田等著. —上海:上海大学出版社,2003.6

ISBN 7-81058-395-6

Ⅰ.软… Ⅱ.刘… Ⅲ.软件质量 Ⅳ.TP311.5

中国版本图书馆CIP数据核字(2003)第052326号

责任编辑:沈荣富　　封面设计:王春杰

责任制作:张继新　　责任校对:张　懿

软件质量评价与保障

刘宗田　潘秋菱　李钢　李心科　等著

上海大学出版社出版发行

(上海市延长路149号　邮政编码200072)

(E—mail:sdcbs@citiz.net　发行热线56331131)

出版人:李顺祺

上大印刷厂印刷　　各地新华书店经销

开本787×1092　1/16　印张11　字数264 000

2003年6月第1版　2003年6月第1次印刷

印数:1～1 100

定价:20.00元

前　言

自计算机软件出现以来，质量一直是困扰人们的难题。随着计算机技术的飞速发展，软件规模和应用领域急剧扩大，质量问题更加突出。由软件质量问题引发的事故悲剧时有发生，因质量不能保障而使项目中途流产的实例屡见不鲜，软件开发和维护的花费远远超出了人们预期的程度。20 世纪 70 年代末，人们再也忍受不住了，终于发出了“软件危机”的呼喊。

实际上，从计算机软件诞生开始，人们就在为软件质量和开发效率的提高而努力，从程序设计语言的发展，到程序设计方法学的研究，人们从数学、语言学、认识论、心理学等各个角度努力探索，这些成果体现在汇编语言、高级程序设计语言、结构化程序设计、程序验证、程序自动生成、程序语义、逻辑型程序设计语言、函数型程序设计语言、面向对象方法学等诸多方面。“软件危机”提出之后，人们开始从工程的角度努力，产生了软件工程学，研究软件开发过程、软件项目管理的规律，以期保证软件质量，提高软件开发效率，降低软件开发成本。

但是，尽管人们做了几十年的努力，但软件质量问题并没有得到根本解决，“软件危机”依然存在。这主要表现在，我们至今还没有一整套较系统的评价软件质量的科学方法，没有监督软件生产过程和中间产品的可靠方法。软件是复杂的具有创造性的产品，其质量标准本身就是不确定的。软件发展的历史不长，飞速发展的技术使得一切都在变化，到目前为止，我们还没有积累出可用来研究软件开发过程规律的足够数据。这一切，是软件质量难以保障的根本原因。

ISO 9001 的制定，标志着人们通过规范软件生成和管理过程而保障质量的思想的实施达到了一定程度，CMM 将这一思想推向了更高的热潮。这无疑是正确的途径。但是，科学的管理必须建立在量化分析的基础上，无法度量的产品或过程是不可控制的。ISO/IEC 和 IEEE 已经认识到这一点，分别制定了软件产品评价标准和软件质量度量方法学标准；CMM 明确地将其第四级定为度量级。只有通过度量，在积累了大量有价值数据的基础上才谈得上优化。

软件度量已经形成了一门学科，目的是测量软件的各方面特性，并且将这些特性的度量值与质量关联。但是，度量什么？如何度量？至今仍然是未能完全解决的问题。度量的目的是评价，这些度量数据是否足以支持评价，评价的标准是什么？标准制定的依据是什么？虽然 ISO 制定了软件质量标准，但计算机软件技术发展日新月异，标准必须发展，评价模型必须更新，如何发展？如何更新？

度量是对软件静态的测量，但软件是在机器上运行的复杂逻辑产品，单靠静态的分析，目前的理论和技术还达不到足以判定软件可靠性和性能的程度，软件测试一直是检查软件可靠性和性能的主要方法。但是，如何测试？如何生成测试数据？如何根据测试结果判定软件的可靠性和性能，仍然有大量的问题需要研究。

在软件开发过程中控制软件质量比产品完成后的检测更有效，这是很明显的道理。但是，如何在软件开发过程中实施控制，这是非常困难的任务。这需要有对过程的规范化描述方法，有对软件中间产品和过程的度量能力，有对资源合理调配的策略，有对整个企业质量保障的支持平台。全面质量管理的思想似乎已经被众多企业所接受，但实际情况并非如此。实际生产中的具体问题、企业员工的个人英雄主义、客户的不理解、企业的短期效益、紧急的项目期限、各种最新的软件开发技术等等，都会对软件全面质量管理的实施过程造成冲击。严格的全面质量管理实施，无论在技术上还是在企业文化上都存在问题。如何解决好这些问题，将是长期而艰巨的任务。

本书是作者对上述问题进行研究所取得成果的系统阐述，向读者介绍了有关概念、原理、理论和技术，提供了实用工具和平台实现技术。希望本书能对软件质量研究者、软件企业和软件项目的管理者、软件开发者、软件项目的委托者和投资者以及有关的行政管理者有所帮助。本书不可能包含有关软件质量与保障的所有问题，而是侧重于作者认为重要的并且在其他已经出版的著作中论述较少的一些问题。例如软件测试，是目前公认的检测软件的重要手段，由于这方面的论述太多了，本书没有收入这部分内容。

本书分为十章。

第一章：软件与软件质量，介绍了软件特点、有关软件质量的基本概念和质量评价与控制综述、软件工程中有关质量保障的一些方法与技术问题，分析了 ISO 9001、CMM、ISO / IEC 12207(IEEE/EIA 12207)、TL 9000、ISO 15504、ISO 9001:2000 版的基本思想。

第二章：软件质量评价，介绍了软件评价的基本原理，对传统的评价模型进行分析与评论，根据软件开发技术和软件度量技术的新发展和新特点，提出了一种新的评价模型，研究了基于回归和人工神经元网络的建立映射的方法。该模型重新定义的软件质量指标，能更有效地与现代度量方法相结合。

第三章：软件度量学，介绍了传统的软件产品度量和面向对象软件的几种度量方法，提出了改进 C & K 度量方法，介绍了作者开发的 C++软件度量工具和 JAVA 软件度量工具。

第四章：软件需求分析与设计度量，介绍了软件功能点度量以及软件分析和设计的中间产品度量，介绍了作者开发的基于 UML 的软件分析与设计度量工具。

第五章：程序分析与程序理解，分析了程序分析和理解方面的研究工作，从不同的侧面对专家分析程序的模型进行了讨论，根据对程序所固有的性质的分析提出了程序分析方法学的一些观点，最后介绍了作者设计的 Java 源程序辅助分析理解系统原型的设计原理和系统结构。

第六章：软件缺点分析和度量，介绍了软件缺点定义，提出了软件缺点度量概念和软件缺点分析与度量技术，介绍了作者开发的软件缺点分析与度量工具。

第七章：软件故障树，介绍了软件故障树定义，提出了基于面向对象软件故障树生成和分析技术，介绍了作者开发的软件故障树工具。

第八章：软件过程技术，回顾了支持软件质量管理的软件过程技术的发展状况，分析了作为过程开发起始的软件过程建模方法，提出了基于 Petri 网的软件过程网(SPNet)，研究了其合理性的定义及性质，讨论了一个实际软件开发企业中的应用实例，最后，采用 XML 语言表述 SPNet，实现了与其他过程建模系统进行交换和互译。

第九章：软件质量管理中的人力资源管理，讨论人力资源的管理方法，主要包括人力资

源建模的关键要素、资源选择的不同方法、动态资源分配的策略和权限控制模型，最后，提出了中小型软件开发企业的详细角色模型 SoftevRM，与资源管理模块相结合，共同完成软件过程开发环境中的资源自动分配和管理。

第十章：基于过程和度量的软件质量保障平台，介绍了作者开发的基于度量的支持软件过程管理平台，该平台可以支持软件开发过程质量控制，辅助软件企业能力成熟。

本项研究工作和本书的编撰工作得到了国家机械工业发展基金“软件故障分析和软件质量评价”（编号：96JA0205）和上海市高等学校科学技术发展基金项目“软件开发质量管理与控制平台研究”（编号：02AZ86）的资助。本项研究还得到了上海亚太计算机信息系统有限公司、全美测评软件系统（北京）有限公司、合肥科大恒星有限公司、上海林果有限公司等多家企业的协助和支持。参加本书编写工作和项目研究工作的有：刘宗田、潘秋菱、李钢、李心科、吴耿锋、袁兆山、许东、邢大红、姜川、胡仁胜、潘飚、何允如、贾亮、徐庆、张为民、强宇、孙志勇、童朝柱、叶震、冯鸿、邵堃、肖苑、李航、程莉莉等。在此，特向为本书付出辛勤劳动和做出有益贡献的所有人员表示衷心的感谢。

由于时间和水平限制，著作中难免有错误和不妥之处，恳请读者指正。

作　者

2002 年 9 月 16 日于上海

目　录

1　软件与软件质量 …… (1)
1.1　软件及其特点 …… (1)
1.2　软件质量 …… (3)
1.2.1　软件质量定义 …… (3)
1.2.2　软件质量管理标准 …… (4)
1.2.3　软件质量评价 …… (6)
1.2.4　软件质量控制 …… (6)
1.2.5　软件质量保障 …… (7)
1.2.6　软件质量代价 …… (7)
1.3　软件工程中的质量保障 …… (8)
1.3.1　程序正确性证明 …… (9)
1.3.2　软件测量学 …… (10)
1.3.3　程序测试 …… (11)
1.3.4　软件可靠性 …… (11)
1.4　质量保障体系 …… (12)
1.4.1　ISO 9000:1994 系列质量保障模式 …… (12)
1.4.2　软件能力成熟度(集成)模型(CMM 和 CMMI) …… (13)
1.4.3　ISO/IEC 12207(IEEE/EIA 12207) …… (14)
1.4.4　TL 9000 …… (14)
1.4.5　ISO 15504 …… (15)
1.4.6　ISO 9001:2000 版的新特点 …… (15)
1.5　小结 …… (17)

2　软件质量评价 …… (18)
2.1　引言 …… (18)
2.2　软件质量评价模型分析与评论 …… (20)
2.3　定义 …… (22)
2.4　一个新的质量评价模型 …… (23)
2.4.1　可用性 …… (23)
2.4.2　可维护性 …… (23)
2.4.3　可重用性 …… (24)
2.4.4　模型结构 …… (24)
2.5　软件测试、度量和评估 …… (24)

2.5.1 测试 …… (24)
2.5.2 度量 …… (25)
2.5.3 评估 …… (25)
2.5.4 质量要素的计算模型 …… (26)
2.6 评价模型的使用 …… (26)
2.6.1 推荐的度量方法 …… (26)
2.6.2 推荐的测试方法 …… (26)
2.6.3 推荐的测量数值到要素值之间的映射函数确定方法 …… (27)
2.7 小结 …… (28)

3 软件度量学 …… (29)
3.1 软件度量学发展历史和现状 …… (29)
3.2 度量方法的评价标准 …… (31)
3.3 面向过程的软件度量方法 …… (33)
3.4 面向对象的软件度量学 …… (34)
3.4.1 面向对象技术综述 …… (35)
3.4.2 面向操作的度量方法 …… (36)
3.4.3 Lorenz 和 Kidd 度量方法 …… (36)
3.4.4 Chen & Liu 度量方法 …… (37)
3.4.5 MOOD 度量方法 …… (38)
3.4.6 Chidamber 和 Kemerer 度量方法 …… (39)
3.5 C++与改进 C & K 软件度量方法 …… (41)
3.6 C++软件度量工具 SMTCPP …… (43)
3.7 度量实验数据分析 …… (44)
3.8 Java 度量工具 …… (47)
3.9 小结 …… (48)

4 软件需求分析与设计度量 …… (49)
4.1 需求分析和设计度量研究现状 …… (49)
4.1.1 需求分析度量研究 …… (49)
4.1.2 设计度量研究 …… (50)
4.2 功能点度量 …… (51)
4.2.1 功能点度量概述 …… (51)
4.2.2 功能点度量方法 …… (51)
4.2.3 功能点度量的应用 …… (53)
4.3 UML 简介 …… (55)
4.3.1 UML 发展历史 …… (55)
4.3.2 UML 的基本概念 …… (55)
4.4 基于 UML 的软件需求分析和设计度量指标 …… (58)

4.5 度量指标分析 …… (62)
4.6 基于 UML 的软件分析与设计度量工具 UMTSA …… (63)
4.6.1 UMTSA 系统功能 …… (63)
4.6.2 UMTSA 系统设计与实现 …… (63)
4.7 实验结果分析 …… (64)
4.8 小结 …… (68)

5 程序分析与程序理解 …… (69)
5.1 程序分析的目的和意义 …… (69)
5.2 有关工作分析 …… (70)
5.3 程序分析的专家模型 …… (72)
5.4 程序分析方法学 …… (73)
5.4.1 程序分析的基础 …… (73)
5.4.2 程序分析的活动 …… (74)
5.4.3 结构程序代数 …… (75)
5.4.4 程序注释 …… (75)
5.5 Java 源程序辅助理解系统原型(HJAUS) …… (75)
5.5.1 设计原理 …… (75)
5.5.2 HJAUS 的体系结构 …… (76)
5.5.3 程序信息库逻辑结构 …… (77)
5.6 HJAUS 实现技术 …… (78)
5.6.1 程序信息元素的自动识别 …… (78)
5.6.2 源程序信息的超文本组织 …… (80)
5.6.3 用户界面设计 …… (81)
5.7 小结 …… (82)

6 软件缺点分析与度量 …… (84)
6.1 软件缺点与软件缺点分析 …… (84)
6.2 面向对象软件缺点分类 …… (86)
6.3 软件缺点分析工具研究 …… (88)
6.3.1 基于语法和语义的分析方法 …… (88)
6.3.2 基于知识的软件缺点分析 …… (88)
6.4 C++软件缺点分析工具的实现 …… (91)
6.4.1 信息提取 …… (91)
6.4.2 缺点分析 …… (92)
6.5 实验与结果分析 …… (94)
6.6 小结 …… (96)
7 软件故障树 …… (98)
7.1 故障树概述 …… (98)

7.1.1 故障树基本概念 …… (99)
7.1.2 故障树在复杂工业系统中的应用 …… (100)
7.1.3 软件故障树研究概况 …… (101)
7.2 面向对象软件故障树模型研究 …… (102)
7.2.1 面向对象软件特征简介 …… (102)
7.2.2 面向对象软件故障树模型构造 …… (103)
7.3 C++软件故障树工具 TOSFT 实现与实验研究 …… (108)
7.3.1 C++软件故障树工具设计思想 …… (108)
7.3.2 C++软件故障树工具主要算法 …… (109)
7.3.3 C++软件故障树工具界面实现 …… (115)
7.3.4 C++软件故障树工具实验分析 …… (115)
7.4 小结 …… (119)

8 软件过程技术 …… (120)
8.1 软件过程 …… (120)
8.2 过程建模语言 …… (122)
8.3 Petri 网概述 …… (125)
8.3.1 经典 Petri 网 …… (126)
8.3.2 Petri 网的基本属性 …… (127)
8.3.3 高级 Petri 网(HLPN) …… (128)
8.4 软件过程网(SPNet) …… (130)
8.5 Petri 网对软件过程技术基本结构的表示 …… (133)
8.6 应用实例 …… (134)
8.7 与其他系统的交互 …… (136)
8.7.1 过程定义语言 …… (136)
8.7.2 XMSPN 的属性 …… (141)
8.8 小结 …… (141)

9 软件质量管理中的人力资源管理 …… (142)
9.1 人力资源建模要素与分析 …… (142)
9.2 资源分配策略 …… (143)
9.2.1 资源划分函数 …… (143)
9.2.2 资源选择的方法 …… (143)
9.3 权限控制模型研究 …… (144)
9.3.1 操作系统和数据库权限控制模型分析 …… (144)
9.3.2 基于安全组的权限控制 …… (144)
9.4 资源建模模块的总体结构 …… (145)
9.5 权限控制模块的设计 …… (145)
9.5.1 安全组和组织单元的区分 …… (146)

9.5.2 工具安全组的设计 …… (146)
9.5.3 过程安全组的设计 …… (146)
9.6 软件开发企业人力资源模型 …… (147)
9.7 小结 …… (153)

10 基于过程和度量的软件质量保障平台 …… (154)
10.1 平台结构 …… (154)
10.2 过程建模工具实现方案 …… (156)
10.3 引擎的实现方案 …… (158)
10.4 监控系统的实现方案 …… (161)
10.5 资源分配模块实现方案 …… (162)
10.6 度量模块实现方案 …… (163)
10.7 客户端应用的实现方案 …… (163)
10.8 软件开发和质量保障平台的现实意义 …… (163)
10.9 小结 …… (164)

1 软件与软件质量

本章主要讨论软件、软件产品与软件质量的基本概念和特点，介绍软件质量的管理标准和软件质量评价基础。

1.1 软件及其特点

按照一般的定义[①]，软件是：① 指令集，即计算机程序，通过执行这个指令集能够提供预期的功能和性能；② 能使计算机程序有效地管理信息的数据结构；③ 描述关于计算机程序的文档，包括设计文档、测试文档、维护文档以及操作和使用说明等。

软件是逻辑产品，具有不同于物理系统的特点：

软件是设计、构思和编写成的，而不是像其他物理产品那样制造的。社会上的人造物基本上可以分为三类：第一类是培育物，包括最原始的农作物和最现代的生化产品；第二类是制造物，从最早的石器到现代最先进的计算机硬件等；第三类是图形文字符号产品，例如小说、乐谱、绘画、文件、电影、电视等。计算机软件属于哪一类呢？从形式上看，它属于第三类，是文字符号产品。但从生产过程角度来看，它又与第二类产品类似，特别类似于第二类产品的设计过程，例如建筑设计、机械设计和化工工艺设计等等。如果把软件生产与机械产品的生产过程类比，区别在于机械产品的设计更规范而软件产品的设计更随意；机械产品的制作过程复杂，并在整个生产过程中所占比重很大，而软件产品的制作过程简单，只是简单的编译和复制过程。有人把软件生产过程与小说、剧本生产相类比，发现程序员与生产小说或剧本的人群有大量的共同点[②]：它们的思想最终都是用某类文本形式表达的，这种表达的结构能确定产品的成功与否。这一类比也很有道理。由于软件产品的这样的生产特点，因此既有与第二类产品和第三类产品类似的质量特点，又有与它们不同的质量特点。相同特点是：它们的高质量都是通过人的活动得到的，换句话说，人的过失造成了产品的缺陷。不同特点是：软件的缺陷完全是由于人的过失造成的；而第二类产品缺陷的产生除人的过失因素之外，还受制造材料的问题、制作工艺以及制作环境等问题的影响；第三类产品的质量含有难以把握和测量的艺术性特征。

软件不能“用坏”。第二类产品的故障产生，除去设计缺陷的原因外，还有产品寿命的因素，也就是正常“用坏”的问题。当然，设计缺陷也与产品部件的寿命关联。比如，大桥的一个铆钉的意外“用坏”，可能导致整座大桥的垮塌；飞机的一个零件的意外“用坏”，可能导致

---

① 郑仁杰. 实用软件工程 . 北京：清华大学出版社，1991

② B. Eckel. Thinking in C++, Prentice Hall, Inc, 1995，或参看，刘宗田等译. C++编程思想. 北京：机械工业出版社，2000

飞机失事。零件的这种意外“用坏”是造成第二类产品故障的主要原因，也是最不容易被发现的设计缺陷或制作缺陷。原则上讲，软件产品不存在“用坏”问题，不论是意外“用坏”还是正常“用坏”，其故障原因完全是由于设计（包括编码）缺陷所引起的。

软件产品的“缺陷隐蔽性”比物理产品的强。所谓“缺陷隐蔽性”是指，在整个产品的生命周期中，产品缺陷能够躲过各个阶段的复审、检查和测试等过程而继续滞留在产品之中的可能程度。软件产品的缺陷隐蔽性主要与软件的复杂性和产品的使用条件的复杂性有关。而物理产品的缺陷隐蔽性主要与零件被“用坏”的情况的复杂性有关。对于软件产品而言，设计缺陷有些在软件的实现阶段和测试阶段得以纠正，但有些会继续残留在产品之中，有的甚至会长期隐藏。而且，实现阶段和维护阶段又会引入新的缺陷。这主要因为软件的输入情况非常复杂，分析和测试只能模仿其中的一部分，实际上“穷举测试”是不可能的。对于物理产品而言，设计缺陷有些在制造阶段得以纠正，但有些会继续残留在产品之中，而且制作过程同样会引入新的缺陷。产品的检查、测试和试运行是发现缺陷的主要手段。虽然物理产品的使用条件的复杂性一般比软件产品简单得多，但由于零件被“用坏”的情况复杂，仍然会有一些缺陷隐藏在产品之中。

软件产品的修改较物理产品容易（但软件产品的故障定位却非常困难）。因为软件是逻辑的字符产品，因此在任何阶段都可以对其修改，包括增加其功能。这是物理产品无法做到的。物理产品因“用坏”而报废，而软件产品因修改维护过多而报废。产品被丢弃的主要原因有二，其一是因为产品过时落后而被淘汰；其二是因不能继续使用而报废。“用坏”是物理产品报废的主要原因，无论是意外的“用坏”还是正常的“用坏”。而软件产品不存在“用坏”问题，理论上讲它不应当有报废的问题。但实际上，正是因为所谓的软件产品“修改容易”的特点，使人们对修改抱有了不切合实际的期望，致使滥用修改和过分地修改，最终导致在修改中引入了更多的缺陷，以至于软件无法维护而报废。另外，由于软件残存缺陷过多，也会导致不得已的过多修改。

在整个产品的生命周期中，就故障率和时间的关系来看，物理产品符合所谓的“浴缸曲线”，如图 1.1 所示，而软件产品的表现将复杂得多，如图 1.2 所示。

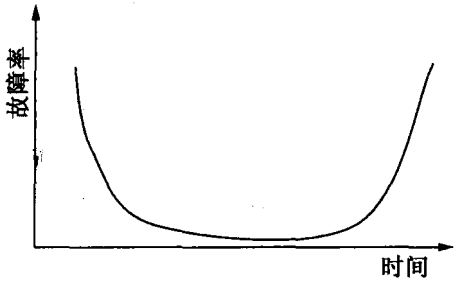

图 1.1　浴缸曲线

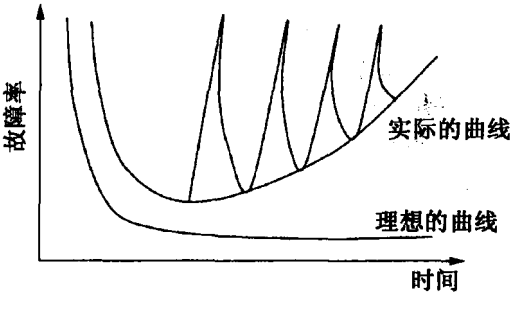

图 1.2　软件故障曲线

物理产品由于其设计和制造问题，以及产品各个零部件之间的“磨合”问题，在产品使用初期，其故障率很高。随着暴露出来的问题被解决，物理产品度过“磨合”期而进入稳定期。在稳定期内，产品的故障率保持在一个较低的水平。稳定期过后，由于产品部件经受了长期的热场、磁场、震动等外界环境的影响及物理特性的老化，其故障率又迅速提高。

软件产品不存在受诸如热场、磁场、震动等外界环境的影响，也不存在“磨合”和“老化”

问题。因此，软件产品理论上的故障率曲线如图 1.2 中的“理想的曲线”。

但实际情况并非如此。由于在整个的软件生命周期中，软件产品将经历多次的变化和维护，而每一次的变化和维护都不可避免地产生新的软件错误，使得软件故障率迅速提高。

1.2 软件质量

随着计算机应用越来越广泛，软件系统渗透到了人类社会的各个角落，成为国民经济、国防和社会日常生活中不可缺少的重要组成部分。软件的作用和地位越发显得重要。因此，软件的质量问题日益成为人们关注的焦点。另一方面，面对不断变化和激烈竞争的市场，产品质量已成为产品开发公司或企业得以保持其长期优势的关键。

实践证明，产品的质量控制是一件很困难的事情，因为它与众多错综复杂的因素相关。而软件产品质量的控制尤为困难。一方面，软件生产的历史不像硬件那样久远，人们对它的认识还很不充分；另一方面，软件的复杂性远远超过硬件，而其直观性却又远远不及硬件产品。

在产品质量控制中，质量评估是产品质量管理的关键，如果没有科学的质量评估标准和方法，就无从有效地管理质量。

1.2.1 软件质量定义

按照 ANSI/IEEE 1983 年的标准，软件质量定义为“与软件产品满足需求所规定的和隐含的能力有关的特征和特性的全体”。具体包括：

- 软件产品中所能满足用户给定需求的全部特性的集合；
- 软件具有所期望的各种属性组合的程度；
- 用户主观得出的软件是否满足其综合期望的程度；
- 决定所用软件在使用中将满足其综合期望程度的软件合成特性。

这一定义重点强调了软件的特性或特征，强调了这些特性或特征与软件需求的吻合程度以及对它们的综合评价值。

一般而言，不同性质和用途的软件，会有不同的特征集合和质量要求。例如实时控制软件和大型联机事务处理软件对于软件的可靠性要求很高。而常规的办公事务软件及管理信息系统(MIS)对于易用性和可移植性要求则比较高。但从各类软件综合起来看，可以列出下列 7 个主要特征：

- 功能性：指软件所实现的功能达到或满足用户需求和设计规范的程度。这里所要强调的是，用户需求不仅包括显性陈述的需求内容，还应包括隐含的需求。
- 可靠性：指软件在指定条件下和特定时间段内，维持其正常性能水准的能力的程度。
- 易使用性：指用户为使用一个软件产品所付出的学习努力以及其他代价的程度。
- 效率：指在规定的条件下，软件功能与所占用资源之间的比值关系。
- 可维护性：当发现错误、运行环境改变或客户需求改变时，程序可被修改的难易程度。
- 可移植性：指将软件从一种环境移植到另一种环境的难易程度。

● 质量管理：指在整个软件生命周期中，实施质量监督和质量控制的配套程度。

从用户角度来评价软件质量，他们主要关心的是：

● 所需求的功能在软件中是否已经被实现？

● 软件是否可靠？

● 软件的效率如何？

● 软件使用是否方便？

● 将软件移植到另一个环境中去是否方便？

● 改变或增添软件功能是否方便？

从软件开发者角度来考察软件质量，他们所关心的重点和用户的目标一致，但在轻重缓急上却有一定的差别。为了保障软件产品的质量，提高软件产品的生产效率，软件开发者一般会更关心软件的结构和软件的可维护性，关心在开发的不同阶段对软件中间结果检测和结果分析是否方便。

从管理者角度来看，他们通常更注重软件的整体质量，而不是某一个具体的质量特征。他们会根据软件应用的类型和商务需要为不同的特征分配权值，同时还要考虑成本、资源消耗、市场占领、开发周期的限制等等，在各个管理要素和质量改进需求之间寻求平衡。

1.2.2 软件质量管理标准

近二十年来，国际、国内各标准化组织在软件质量管理方面做了大量工作，制定了大量与软件质量有关的标准(可参看注①)。这些标准有：

ISO 8402:1994 质量管理和质量保证——术语；

ISO 9000.1:1993 质量管理和质量保证标准(第一部分)——选择和使用指南；

ISO 9000.2:1993 质量管理和质量保证标准(第二部分)——ISO 9001、ISO 9002 和 ISO 9003的实施通用指南；

ISO 9000-3—1997 质量管理和质量保证标准. 第三部分：ISO 9001—1994 计算机软件的开发、供给、安装和维护应用指南；

ISO 9001:1994 质量体系——设计、开发、生产、安装和服务的质量保证模式；

ISO 9002:1994 质量体系——生产、安装和服务的质量保证模式；

ISO 9003:1994 质量体系——最终检验和试验的质量保证模式；

ISO 9004.1 质量管理和质量体系要素——第一部分：指南，该标准是通用性指南；

ISO 9004.2 质量管理和质量体系要素——第二部分：服务指南，该标准是针对服务业质量管理的；

ISO 9004.4 质量管理和质量体系要素——第四部分：质量改进指南；

ISO 10005 质量计划指南；

ISO 10011.1 质量体系审核指南(第一部分)——审核；

ISO 10011.3 质量体系审核指南(第三部分)——审核工作管理；

ISO 10013 质量手册编制指南；

ISO/IEC TR 9294—1990 信息技术. 软件管理导则；

---

① http://www.cssn.net.cn/vercha.htm

ISO/IEC 9126:1991 信息技术—软件产品评价—质量特征及其使用指南；

ISO/IEC 12207:1995 信息技术—软件生存期过程；

ISO/IEC TR15504:1997 软件过程改进与能力测定；

ISO/IEC 14598-3—2000 软件工程. 产品评价. 第三部分：开发者开发过程；

IEC 61713—2000 软件整个使用期的使用可靠性. 应用指南；

IEC 61014—1989 可靠性增长程序；

IEC 60880-2—2000 核电站安全系统用计算机软件. 第二部分：使用软件工具和预先开发的软件对由常规失误导致的安全防护的软件情况；

IEC 60300-3-6—1997 可靠性管理. 第三部分：应用指南. 第六节：软件样品的可靠性；

IEEE 982.2—1988 可靠软件开发方法的 IEEE 标准词典使用指南；

IEEE 982.1—1988 可靠软件开发方法的标准字典；

IEEE 830—1998 软件要求规范的规程；

IEEE 829—1998 软件测试文件编制；

IEEE 828—1998 软件配置管理方案；

IEEE 730.1—1995 软件保证设计导则；

IEEE 730—1998 软件质量保证计划；

IEEE 1298—1992 软件质量管理系统. 第一部分：要求；

IEEE 1228—1994 计算机软件安全方案；

IEEE 1219—1998 软件维护；

IEEE 1074—1997 开发软件寿命周期的处理；

IEEE Std 1061：1992 IEEE 软件质量度量方法学标准；

IEEE 1063—1987 软件用户文件编制；

IEEE 1062—1998 软件获得规程；

IEEE 1028—1997 软件的复审；

IEEE 1016—1998 软件设计描述的推荐规程；

IEEE 1012a—1998 软件鉴定和认证用 IEEE 标准的补充：IEEE/EIA 12207.1—1997 的内容映射；

IEEE 1012—1998 软件鉴定和认证；

IEEE 1008—1987 软件单元测试；

IEEE 610.12—1990 软件工程术语集；

IEEE 1002—1987 软件工程标准分类学；

GB/T 9386—1988 计算机软件测试文件编制规范；

GB/T 9385—1988 计算机软件需求说明编制指南；

GB/T 8567—1988 计算机软件产品开发文件编制指南；

GB/T 8566—1995 信息技术软件生存期过程；

GB/T 19000.3—2001 质量管理和质量保证标准．第三部分：GB/T 19001 在计算机软件开发、供应、安装和维护中的使用指南；

GB/T 18221—2000 信息技术程序设计语言环境与系统软件接口独立于语言的数据

类型；

GB/T 17544—1998 信息技术软件包质量要求和测试；

GB/T 16704—1996 计算机软件著作权登记文件格式；

GB/T 16680—1996 软件文档管理指南；

GB/T 16260—1996 信息技术软件产品评价质量特性及其使用指南；

GB/T 15853—1995 软件支持环境；

GB/T 15538—1995 软件工程标准分类法；

GB/T 15532—1995 计算机软件单元测试；

GB/T 14394—1993 计算机软件可靠性和可维护性；

GB/T 14079—1993 软件维护指南；

GB/T 13702—1992 计算机软件分类与代码；

GB/T 13423—1992 工业控制用软件评定准则；

GB/T 12505—1990 计算机软件配置管理计划规范；

GB/T 12504—1990 计算机软件质量保证计划规范；

GB/T 11457—1995 软件工程术语；

CMM 1987 美国卡内基-梅隆大学软件工程研究所 SEI 提出的“软件过程能力成熟度模型(Capability Maturity Model)”。

1.2.3 软件质量评价

就一般意义而言，“质量”一词在美国 Heritage 词典中定义为“事物的一个特性或属性”。作为一个物品的属性，质量涉及到一些可测量的特性，即在某些方面能够与已知标准进行比较的属性。例如长度、颜色、电特性、延展性等。但是，软件作为一种智力产品和逻辑系统，对其属性的刻画要困难得多。

程序有一些性质是可以测量的。这些性质包括诸如回路复杂性、内聚性、功能点数、代码行数等以及众多的其他性质。当我们基于可测量特性检查一个事物时，会遇到两种质量问题：设计质量和一致性质量。

设计质量涉及到设计者所指明的产品特性。材料、公差和性能规范的级别都与设计质量有关。如果产品根据指明的规范制造，如高级别的材料、紧密公差和高性能规范被指明时，它的设计质量就得以提高。

一致性质量是指设计规范在以后的生产制造过程中被遵守的程度。一致性程度越高，一致性质量就越高。在软件开发中设计质量包括需求分析、规范说明和系统设计。一致性质量主要集中在实现上。如果实现遵从设计，并且结果系统满足它的需求和性能目标，就可以获得较高的一致性质量。

1.2.4 软件质量控制

软件质量控制是指软件开发周期内所进行的一系列审查、评价、测量等活动，其目的是使每个过程产品能够满足所提出的要求。软件质量控制包括对产品创造过程的反馈循环。当产品不满足它们的规范说明时，测量和反馈的结合使得我们能调整产品的创造过程。

软件质量控制活动的形式大致可以分为自动、手工和自动工具与人工交互相结合三种。

关于质量控制的一个关键概念是，所有的过程产品具有可定义的、可测量的规范说明。根据这个规范说明，可以通过比较、计算等方法产生每个过程产品的有关质量控制的输出信息，并将输出信息反馈给有关的过程。这种反馈循环对于缺陷的最小化来说是基本的保证。

1.2.5 软件质量保障

软件质量保障(SQA)是应用于整个软件过程的保护活动。SQA 包括：① 质量管理方法；② 有效的软件工程技术(方法和工具)；③ 应用于整个软件过程的形式化技术评论；④ 多等级测试策略；⑤ 软件文档以及对软件进行改变和维护的控制和约束；⑥ 确保遵照软件开发标准的过程；⑦ 测量和报告机制。

软件质量保障包括管理的审核和报告功能。软件质量保障的目的是以有关产品质量的基本数据为依据进行质量管理，从而保证产品质量不断地朝着其质量目标迈进。软件质量保障活动提供大量有关质量的基本数据，不过，根据这些数据所指出的质量问题必须通过质量管理来解决。

1.2.6 软件质量代价

软件质量代价是指为提高产品质量或实现质量目标而进行的一系列活动所需花费的总和。软件质量代价为研究质量与花费之间的关系提供了基准，为质量的花费提供了可以比较的标准。这个标准一般都以美元为单位。一旦有了软件质量代价的标准，就有了确定何时何处可能进行软件开发中过程改进的依据。

质量代价可以分为预防性代价、评价性代价和故障性代价。

预防性代价包括：

(1) 质量计划；

(2) 形式技术评论；

(3) 测试设备；

(4) 培训。

评价性代价指每个过程产品首次得到确认而必须进行的一系列检查、测试等活动所需的花费。评价性代价包括：

- 过程内和过程间检查；
- 设备标度和管理；
- 测试。

故障性代价是在产品销售给客户前后，发现产品缺陷并予以纠正所付出的代价。故障性代价可以分为内部缺陷代价和外部缺陷代价。内部缺陷代价是在产品投放前发现产品的缺陷并予以纠正所付出的代价，它包括：

- 返工；
- 修理；
- 故障模型分析。

外部缺陷代价是在产品投放市场后发现产品缺陷并予以纠正所付出的代价，它包括：

- 对产品故障的处理；
- 产品返回和更换；