tware Verification and Validation for Practitioners and Managers Second Edition

国外IT精品丛书

软件验证与确认的最佳管理方法

Steven R. Rakitin 获得电气工程学士学位和计算机科学硕士学位，具有25年以上的工作经验。他被美国质量学会认定为软件质量认证工程师和认证质量审核员。

〔美〕 Steven R. Rakitin 著
于秀山 包晓露 焦 跃 等译

Artech House

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Software Verification and Validation for Practitioners and Managers Second Edition

软件验证与确认的最佳管理方法

〔美〕 Steven R. Rakitin 著

于秀山 包晓露 焦 跃 等译

電子工業出版社

Publishing House of Electronics Industry

北京 · BEIJING

内 容 提 要

本书分四部分向计算机专业管理人员讲述软件验证与确认的方法。详细介绍了软件开发过程、软件验证活动、软件确认活动、可预测的软件开发等内容。帮助专业人士有效地对软件开发过程进行管理，不断地提高软件质量。

本书结构清晰、内容丰富，适用于软件开发人员、工程技术人员、管理人员。也可作为高等院校计算机专业师生的参考书。

图书在版编目（CIP）数据

软件验证与确认的最佳管理方法/（美）拉克汀（Rakitin, S.R.）著；于秀山等译. - 北京：电子工业出版社，2002.3

书名原文：Software Verification and Validation for Practitioners and Managers Second Edition

ISBN 7-5053-7533-4

Ⅰ. 软… Ⅱ. ①拉… ②于… Ⅲ. 软件工程 Ⅳ. TP311.5

中国版本图书馆CIP数据核字（2002）第016249号

责任编辑：贺玉寅

印　　刷：北京天竺颖华印刷厂

出版发行：电子工业出版社　http://www.phei.com.cn

北京市海淀区万寿路173信箱　邮编：100036

北京市海淀区翠微东里甲2号　邮编：100036

经　　销：各地新华书店

开　　本：787×1092　1/16　印张：18　字数：380千字

版　　次：2002年3月第1版　2002年3月第1次印刷

定　　价：28.00元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换，若书店售缺，请与本社发行部联系。联系电话：（010）68279077

前　言

本书介绍一系列基本软件验证与确认活动。根据在几个工业部门的公司工作的切身体会，我发现一些基本的软件验证与确认活动没有得到很好的理解，并且没有得到合理的应用。例如，在我曾经工作过的几家公司里迄今没有软件质量保证小组，少数公司甚至没有落实基本的配置管理方法。另外，在一些拥有软件质量保证的组织机构内，我发现各组织机构的有效性（甚至同一组织机构内各项目的有效性）差别巨大。尽管我的观察不是基于统计学上大量的样本，但我认为自己的经验相当准确地反映了整个工业界的情况。实际上，Yourdon[1]最近报告说，尽管软件质量总体上有所改进，但生产最佳软件的公司与生产最劣软件的公司之间的差距在过去10年里显著加大。

作为一名顾问，我常被公司请去帮助解决“质量问题”。调查过一些有此类问题的委托人后，我发现了一个共性的主题——他们都是以不可预测的方式运转的。例如，这些组织不可能确定大的事件何时发生，例如代码冻结或产品的第一个用户（在某些情况下，甚至不能够知道他们是否会发生）。因为这些组织不具有可预测能力，所以无法预计产品的上市时间，开发与质量保证部门无法有效利用昂贵且稀有的资源，因而软件验证与确认活动几乎无法发挥其应有的效能。

Len Race是一名顾问，也是作者的朋友。他注意到，为了使商家效益更佳，他们必须学会以更长远的可预测的方式运转。我认为这是个新发现。不佳实绩与不可预测的运转状态之间的关系显而易见。不可预测的组织存在如下现象：

- 他们一贯承诺多，兑现少。
- 他们低估所承担的工作，并且忽略几乎每一项计划安排。
- 组织内人员的目标与整个商业目标不一致。
- 整个组织内缺乏责任制。
- 缺乏采用“最佳做法”的意识。即使有书面程序，但一贯不遵照执行。
- 员工们的看法是，我们从来没有时间做对事情，但我们总有时间重做事情。

当我帮助公司解决质量问题时，我都是从首席执行官着手，并询问他：“如何考核贵公司员工的实绩？”通过查看个人的实绩计划，你会了解为什么该组织以其独特方式运转。一个必然的情况是，当查看有质量问题的公司的实绩时，很难发现“质量”这两个字。既然运转状态与如何考核员工直接挂钩，为什么还对

那些组织存在“质量问题”大惊小怪呢？

一旦找出不佳实绩与不可预测的运转状态两者之间的关系，我意识到：

1. 管理部门中既存在问题又有解决办法。
2. 为了提高软件验证与确认方法的有效性，各类组织都必须学会以更加可预测的方式运转。

显然，管理部门必须在帮助组织实现以可预测的方式运转方面起领导作用。因此，对本书的书名做了更改，把管理部门也包括进去。本书涉及管理部门为帮助其组织以更加可预测的方式运转可能采取的具体措施。

本书的读者对象

从本书的书名可以看出，本书面向两个读者群，即专业人员和管理人员。专业人员包括软件质量保证工程师、软件工程师、以及需要基本了解软件验证与确认方法的项目经理。

令人遗憾的是，学校对软件验证与确认方面的正规培训甚少，其结果是，许多软件质量专业人员具有的技能与生产高质量软件所必需的技能之间存在差距。本书的读者对象是那些负责软件验证与确认工作但接受这方面培训不多的人士。第一部分到第三部分内容适合于专业人员。

管理人员包括软件质量保证经理、开发经理、项目经理、负责工程设计和开发的副总经理、质量董事和首席执行官。管理部门通过提供实现商业目标所必需的领导以及确保员工实绩考核方式与商业目标一致，有能力改变其组织的运转状态。从事管理工作的每一位员工，从首席执行官到生产第一线的经理都必须认识到，当组织变得更加可预测时，实现商业目标则容易多了。本书第四部分为各级经理和执行人员提供了具体策略，他们可用这些策略帮助其组织以更加可预测的方式运转。

验证与确认适用的软件

本书所讲述的软件验证与确认活动适合于各种各样的软件、各种各样的产品以及各种各样的产业部门。可以用另一个问题作为这个问题的最佳答案：“有开发高质量软件并准时交付软件的强制性的商业理由吗？”如果答案是肯定的，那么许多软件验证与确认活动也是适用的。

本书的编排结构和第二版本增加的新内容

第一部分包括软件开发引言和软件开发过程概述。第2章介绍几个软件开发生存周期模型。第3章阐述将软件开发过程文档化的重要性。第4章讨论软件验证与确认活动的经济动机。

第1章增加了有关生存周期模型国际标准——ISO 12207方面的论述。第2章涉及瑞理统一的过程方面的信息。第一部分采用了软件标准的最新参考文献。

第二部分概述了软件验证活动。第5章和第6章（以及附录A到附录D）详细介绍了正规的审查过程。第7章侧重于验证测量。第8章介绍配置管理。

第6章对内容稍做整理以删除一些冗余信息。第二部分采用了软件标准的最新参考文献。

第三部分概述软件确认活动。第9章介绍软件测试工作。第10章介绍测试度量。第11章介绍软件可靠性增长。

第9章是重新撰写的一章，详细介绍了各种测试类型，包括并发测试/开发模型、测试计划和测试计划评价技术。第10章也做了修改，专门侧重于确认测量。

第四部分是新内容，它侧重于为管理部门提供具体策略，这些策略可帮助组织以更加可预测的方式运转，从而显著提高软件验证与确认活动的有效性。第12章介绍入门知识和经济动机。第13章讨论质量、特性和进度方面的平衡问题。第14章介绍黄色粘贴法——一种估算任务和确立真实进度的方法。第15章讨论人员需求、过程和产品的平衡问题。第16章讨论管理承诺和风险的方法。

除第四部分外，还增加了5个新附录。

参考文献

[1] Yourdon, E., Rise and Resurrection of the American Programmer, Upper Saddle River, NJ: Prentice-Hall PTR, 1998。

译 者 序

随着信息技术的发展，软件的应用越来越广泛，人们对软件的质量越来越关注。作为软件质量重要保证手段之一的软件验证与确认应运而生。

本书根据作者多年从事软件质量认证工作的经验，从理论与实践的结合上全面、系统地阐述了如何开展软件验证和确认工作，可帮助专业技术人员和管理部门经济、有效地对软件开发过程进行管理，从而提高软件的质量。

全书共分4个部分和15个附录，分别介绍了软件开发生存周期模型、软件验证活动、软件确认活动以及具体策略；15个附录详细给出了软件开发最佳实践、审查过程、文档、各种检查表等范例。书中既包括适合于专业人员的正式审查过程、有效的测试方法、配置管理和过程改进测量的内容，也包括适合于管理部门的可预测的软件开发、准确估计与进度安排方法、质量、特性、进度的平衡方法等关键性内容。

本书由于秀山、包晓露、焦跃、张国卿、贺玉寅、胡兢玉、杨铃萍、赵静、于乐山、贺星、于明、沈涛、邹彤、赵红梅、方文刚、宋刚翻译，全书由张燕虹审校。

译者在翻译过程中，除对原文个别文字错误做了相应更正外，力求信、达、雅，但由于水平有限且时间仓促，错误与不妥之处在所难免，恳请广大读者谅解，并欢迎批评指正。

致　谢

可预测的软件开发的想法是与我的朋友、同事Len Rance讨论时产生的。他对商业过程的洞察力和“做事”能力简直令人不可思议。

我要向Artech House出版社的全体职员致谢。感谢Tim Pitts、Ruth Young、Judi Stone、Jen Kelland以及其他人在整个出版过程中对我的帮助。他们给了我恒心和勇气。

Steven R. Rakitin
Upton, Massacbusetts
2001年7月

作 者 简 介

Steven R. Rakitin获得美国东北大学的电气工程理科学士学位和伦斯勒工业学院的计算机科学理科硕士学位。他在很多工业部门担任软件工程师和软件质量专业人员，具有25年以上的工作经验。他作为一名软件质量工程师和质量审查员，得到美国质量协会的认可。他是美国质量协会和IEEE计算机学会的会员。作为软件质量咨询公司的总经理，Steven R. Rakitin与立志于创建更加可预测的软件开发过程的公司一起奋斗。如果想了解更详细的信息，请访问http://www.swqual.com或通过info@swqual.com与作者联系。

目　录

第一部分　引　言

第一部分旨在为众多软件组织提供其发现自身处境的背景。第1章对所谓的软件危机进行了综述，并提及关于软件危机已经发生和正在发生的事情。第1章还介绍了使管理部门和执行官员懂得软件开发本质的专题。第四部分对该专题进行了深入讨论。

第2章讨论了几个生存周期模型。目的是提供一个整体框架，以便于理解如何将软件验证和确认活动纳入到软件开发过程中。

第3章讨论了将软件开发过程文档化的重要性。

与任何商业活动一样，验证和确认活动必须从经济的角度衡量。第4章以软件验证和确认的经济缘由结束了第一部分内容。

第1章 软件透视

人们依赖于软件，但有时也毁于软件。有些软件故障令人烦恼，而有些软件故障却是灾难性的。技术带来某种风险早已不是新闻。在系统中增加软件可以使系统提供的服务更便宜、更易用、更易修改，但却不会使系统更可靠。[1]

20世纪是先进的硬件和奇特的软件技术迅猛发展的年代。从植入起搏器到火星探测器，在过去的半个世纪里，硬件与软件技术的结合取得了一些最为辉煌的成就，同时也导致了令人震惊的失败。

然而，人们对先进技术产品近乎贪婪的胃口超出了生产产品的能力了吗？如果是这样的话，它是怎样发生的并且软件工业如何才能减轻这种状况造成的后果呢？

1.1 软件危机

在20世纪70年代中期首次出现了“软件危机”一词。所谓危机是认为用当时的软件开发技术开发大型、复杂的软件系统已经超过了我们的能力。

正是在20世纪70年代中期，软件维护活动的费用首次超过了开发新软件的费用。也正是在那个年代，我们看到了在后几年变得十分严重的趋势苗头：硬件费用急剧下降而软件费用持续增长，许多项目由于软件不断庞大而失败。

当时，许多人认为我们只要有更好的编程语言，就可以从危机中解脱出来。因此，诸如PL/1，Jovial及APL编程语言日益流行。但仍然是失败连连。

例如，用于F-16的导航软件中的错误使其在飞越赤道时引起飞机翻转。1981年，对航天飞机定时软件的极小改变引起发射失败，即使世界顶级软件工程组对其进行了几千小时的测试。更遗憾的是，至少有两人死于Therac-25线性加速器中软件错误引起的射线过量[2]。

为试图避免将英语书写的需求错误地转换成程序而引发的问题，对用形式化语言说明需求展开了许多研究。

形式化规格说明语言（如HAL/S）的开发使建立基于自然语言的规格说明成

为可能。以形式化语言开发需求，然后将形式化的需求送给编译器，编译器将形式化说明直接转换成传统的编程语言。以前许多为航天飞机开发的软件都是用HAL/S编写的。

高度结构化的多任务编程语言（如Modula和Ada）被开发用于处理那些具有实时、多任务需求的应用。

实际上，编程语言对于整体软件可靠性的影响比其他因素相对要小。诸如PL/1、Jovial和APL这样的编程语言并不广泛用于商业应用，现在最广泛使用的一些语言包括C、C++和COBOL。Web应用常用Java、Visual Basic和HTML开发。这些语言都不是为解决软件可靠性问题而开发的。

1.2 虚幻的银弹

直到1985年，软件工程被公认为是一个有关软件的工程学科。许多公司意识到如果要保持竞争力，就必须对软件开发过程进行重大改进。到20世纪80年代中期，软件已经成为产值超过3千亿美元的工业。

硬件费用不断大幅度地下降。许多强大的新型工作站被开发。这些工作站以及由其组成的网络为商业化的计算机辅助软件工程（CASE）工具提供了平台。计算机辅助软件工程工具实现了特定的软件开发过程（如Yourdon的结构化设计、WardMellor或Hatley-Pirbhai）。这些工具为软件工程师提供了图形化的方式进行软件设计，使得软件设计易于维护、交叉检查，更重要地是易于理解。

许多人认为计算机辅助软件工程工具是拯救软件工业摆脱软件危机的银弹。但许多公司耗费大量资金购置不常用的工具结果怎样呢？这些工具实现的过程常常是不易理解的或是与公司的软件设计过程不一致。艰苦的历程告诉我们世上并不存在什么银弹[3]。

1.3 解决危机的其他尝试

还有一些用于解决软件危机的其他尝试，其中多数尝试或多或少起到一些作用。

1.3.1 正确性形式化证明

正确性形式化证明是用数学证明程序正确性的方法。将程序作为数学对象，这样可用数学的方法证明程序是正确的。由于编程语言是基于严格的语法和语义规则的，因此这种方法是可行的。

数学家们对这种方法最感兴趣，比较适合学术研究。然而其实用价值却有限，其原因是，直到编码完成后才可以应用形式化证明。到那时，通常是太迟了，而且对大型程序进行证明也是一件非常困难的事情。

1.3.2 独立的验证与确认

NASA以及美国国防部（DoD）率先通过第三方机构评审关键项目承制方的软件开发工作。独立的验证和确认（IV&V）直接向用户负责，与承制方一样承担许多任务，如需求分析、需求跟踪、体系结构评审、设计评审、代码审查以及确认测试。

独立的验证和确认非常有效，但用于所有应用开销太大，通常用于最关键的应用，如飞行控制软件及植入起搏器软件等。

1.3.3 软件质量保证

对大多数软件而言，不可能承受独立的验证和确认承包商的费用。许多公司建立了软件质量保证（SQA）部门作为内部独立的验证和确认小组。软件质量保证小组执行许多与独立的验证和确认一样的活动。

软件质量保证已被广泛接受，它以一种实际的、合理的效费比方式提高软件质量。实践证明，内部软件质量保证小组可以有效地提高质量，除测试外他还兼有更广泛的评审职能。然而，如何实现跨公司的软件质量保证仍是个悬而未决的问题。

1.3.4 净室过程

从前在IBM联邦系统部供职的Harlan Mills博士和一个软件过程幻想家共同开发了净室过程。该过程[4]由具有统计过程控制（SPC）的形式化验证程序组成。这种方法首先强调的是用正确性的数学证明而不是调试进行缺陷预防。平均故障间隔时间（MTBF）用于软件质量的度量。

净室过程相对而言是比较新的概念，还没有被广泛接受。该方法要求在软件开发管理和技术方面要有重大的改变（特别是将统计过程控制知识应用到软件中时），这一点进一步影响了净室过程的普及。

1.4 理解软件的本质

在许多软件组织的管理层中普遍存在这样的态度：“我们按照自己的方式开发软件很成功，为什么还要改变呢？况且质量不能当产品销售，而特性却能。”

你可能记得在20世纪60年代至20世纪70年代汽车工业的执行官员们对于其产品也持有类似的态度。我们也知道当时发生了什么。

> 获得较高的软件质量等级是公司可以遵循的最有效的商业策略之一。高质量意味着用户满意，员工有信心，开销及进度可控，具有竞争力。没有什么能比这更深入人心的了。反之，低劣的质量耗尽资金，挫伤开发者的积极性，惹恼用户或使用户敬而远之，最糟糕的是公司有可能被起诉、破产或同时遭受两种厄运。[5]

提高软件开发的生产率和质量已势在必行，特别是在那些诸如美国软件开发业那样获得较高效费比的国家更是如此。例如，据Ed Yourdon报道，在印度班加罗尔的摩托罗拉分部是首批达到软件工程研究所（SEI）推出的能力成熟度模型（CMM℠）第5级的企业之一。他进一步报道说："在世界的每一个角落都存在着好或坏的软件企业，像印度一些国家中许多雄心勃勃的软件企业都渴望其高质量的工作得到国际的认可，以使他们更具竞争力。"[6]

在典型的软件企业中，大多数高级管理部门并不完全懂得如何开发好的软件。更重要的是，大多数高级管理部门和经理们不了解项目组如何看待他们所生产产品的质量。正如DeMarco和Lister所描述的：

> 我们的经理们试图将质量看成是产品的另一个属性，是一种可以根据市场需要程度来提供的东西。
>
> 另一方面，生产者对质量的看法则大不相同。由于他们的自尊心很大程度上依赖于产品的质量，他们试图将质量标准强加于自己。最起码他们要达到过去所获得的最佳质量。这一直是比市场需求要高的标准，但他们愿意为之努力。[7]

作为经理，我们必须重新认识产品质量及产品上市时间的重要性。质量不仅对于用户十分重要，而且对于所有的员工也十分重要。只有当产品的质量确实过关时，才会有好的销售量！

1.5 软件过程改进创新

目前在改进软件质量和可靠性方面已取得了一些重大进展。一些瞄准改进软件开发过程的创新已见成效。下面介绍一些这样的创新。