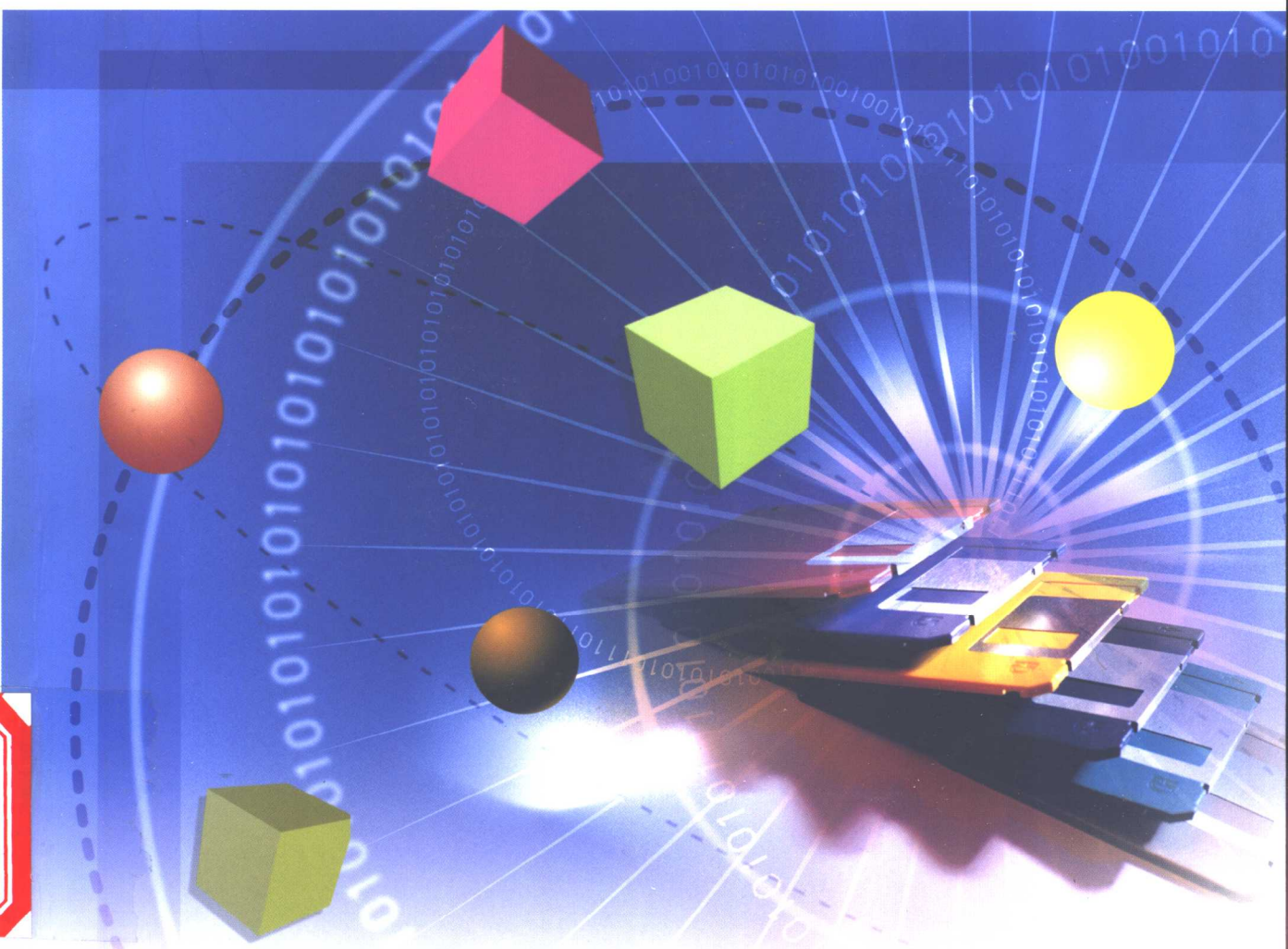
软件能力成熟度模型集成（CMMI）培训教程

罗运模 等 编著



清华大学出版社

软件能力成熟度模型集成(CMMI)培训教程

罗运模 等 编著

清华大学出版社
北京

内 容 简 介

本书从进行软件过程改进和通过软件能力成熟度评估的角度引导读者阅读和理解 CMMI 模型，以便进行软件过程的改进工作，并顺利通过 CMMI 的评估。为便于国家行业标准的实施，本书结合了国家信息产业部于 2001 年 4 月发布的《中华人民共和国电子行业标准(SJ/T 11235－2001)——软件能力成熟度模型》和《中华人民共和国电子行业标准(SJ/T 11234－2001)——软件过程能力评估模型》两个行业标准。

本书首先简要介绍了 CMM 与 CMMI 以及它们的差别，然后重点介绍了 CMMI 模型的框架、CMMI 过程域解读示例、CMMI 的评估方法、CMMI 软件能力成熟度等级过程域等内容，最后介绍了 CMMI－SW 模型的受管理级、已定义级、定量管理级、持续优化级的全部过程域的具体内容。

本书可以作为软件技术人员掌握 CMMI 的基本知识和核心内容的自学教材，亦可以作为软件组织实施软件过程改进和进行软件能力成熟度评估的指导文献。本书还可以作为软件学院的 CMMI 课程的教材或参考书。

图书在版编目(CIP)数据

软件能力成熟度模型集成(CMMI)培训教程/罗运模等编著. —北京：清华大学出版社，2003.10

ISBN 7-302-07303-1

Ⅰ. 软… Ⅱ. 罗… Ⅲ. 软件开发－技术培训－教材 Ⅳ. TP311.52

中国版本图书馆 CIP 数据核字(2003)第 085842 号

出 版 者：清华大学出版社　　地　　址：北京清华大学学研大厦
http://www.tup.com.cn　　邮　　编：100084
社 总 机：010-62770175　　客户服务：010－62776969
责任编辑：宋　韬
封面设计：付剑飞
印 装 者：北京市清华园胶印厂
发 行 者：新华书店总店北京发行所
开　　本：185×260　印张：24.5　字数：566 千字
版　　次：2003 年 10 月第 1 版　2003 年 10 月第 1 次印刷
书　　号：ISBN 7-302-07303-1/TP·5303
印　　数：1～4000
定　　价：39.00 元

前　言

软件能力成熟度模型(Capability Maturity Model For Software,CMM)是由美国卡内基梅隆大学软件工程研究所(Software Engineering Institute,SEI)受美国国防部委托研究制订,于1991年首先在美国应用,随后在全世界推广实施的一种软件能力成熟度评估标准,主要用于指导软件开发过程的改进和软件开发能力的评估。

CMM在全世界的应用推动了软件企业的软件过程改进和软件管理能力的提高,从而极大地提高了软件项目的控制能力和软件产品的质量,促进了全世界软件产业的健康发展。

CMM等级的评估为软件产品发包方选择承包商提供了便利。没有CMM等级的评估,不可能承接美国国防部的软件项目。此外,越来越多的软件产品发包方根据CMM等级的评估结果来选承包商。

2000年,国务院发布了《鼓励软件产业和集成电路产业发展的若干政策》,由此我国软件产业进入了飞速发展时期,CMM等级的评估也已成为我国软件产业走向世界的重要条件之一。

CMM的应用虽然促进了全世界软件产业的发展,但在实际应用中也发现了一些问题和缺陷。由此,美国卡内基梅隆大学软件工程研究所于1999年提出了新的软件能力成熟度模型——软件能力成熟度集成模型(Capability Maturity Model Integration For Software,CMMI)。CMMI是CMM的改进模型,并且从2001年起,已经开始逐步代替CMM。有消息报道,到2005年美国卡内基梅隆大学软件工程研究所将停止CMM评估的备案工作,转而全力做好CMMI的推广和备案工作。

和CMM模型一样,CMMI模型也是关于软件过程改进的指导性文献,但它仅仅告诉软件组织进行软件过程改进需要做哪些事情,而没有告诉软件组织怎样做这些事情。因此,怎样做到CMMI模型中所规定的目标,需要软件组织发挥自己的优势来达到这些目标。

事实上,软件开发的情况各式各样,很难也不可能有一个标准的到达CMMI所规定的目标的具体方法。读者可以参考很多已经成功地实施了CMM或CMMI认证的软件专家或软件组织所编写的经验总结书籍。例如,可以参考主要参考文献中的文献8和10。当然,除此之外,也还有许多的资料可供参考。

在工程领域,组织的质量和生产率依赖于3个主要的因素:过程、人员和技术。在一些大型系统的开发领域,随着技术的不断进步和人们素质的提高,过程因素逐渐成为制约产品质量和生产效率的瓶颈。因此,在开发组织中进行过程改进,进而增强其过程能力成为了开发组织必须要做的一项工作。

在我国引进CMM的初期,有一个非常令人困惑的问题,这就是好像CMM的目标就是花费10万元或100多万元人民币来进行CMM评估,似乎CMM就是为评估而创建的。当时,从国外请来讲座的专家们也是仅讲CMM的评估,这更加深了人们的误解。事实并不是这样。CMM和CMMI的目的其实是为了帮助软件组织进行软件过程改进,提

高软件过程能力、软件产品质量和软件开发效率,从而提高软件组织自身的国内和国际竞争力。CMM或CMMI的评估仅仅是对软件组织的过程改进成果进行实事求是的鉴定诊断,判断经过程改进后的软件组织的软件过程能力或软件能力成熟度等级。因此,CMM或CMMI的评估仅仅是促进软件过程改进的手段,评估本身并不是目的。

CMMI模型不但子模型众多,而且是一个不断修改的软件过程改进模型,它不断变化。例如,CMMI的最新版本是2002年公布的。软件组织如果要进行软件过程改进,并希望进行CMMI模型的评估时,要特别注意CMMI模型的最新版本。

本书是关于CMMI模型阅读和理解的指导性文献,而不是关于过程改进的指导文献。CMMI模型是一种关于软件过程改进的标准化文献,一般读者难以阅读和理解。一位从事标准化工作30多年的专家曾这样评价CMMI模型:尽管我从事标准化工作几十年,但我从未见过像CMMI模型这样的类似天书一样的标准。本书就试图将此天书拉回到地面上来,希望这个目标能够真正实现。

本书介绍CMMI的发展历程、基本框架结构、CMMI过程域的基本结构、CMMI的评估和CMMI的成熟度等级等基本内容,并以CMMI-SW模型为例介绍了CMMI软件能力程度模型的过程域的特定目标和特定实践。这些特定目标和特定实践是CMMI模型中最为重要的部分。

本书分两大部分共9章。第一部分包括前5章,是关于阅读和理解CMMI模型的内容;第二部分包括后4章,是CMMI-SW模型的阶段式表示法的内容。

第1章介绍CMM与CMMI的相同之处和不同之处,主要包括软件产业的发展历程,CMM的引入,CMMI的提出,CMMI奠定中国软件能力成熟度模型的行业标准和CMM与CMMI的比较等内容。

第2章介绍CMMI的框架结构,主要包括CMMI过程域的能力等级特征,CMMI的成熟度等级特征,CMMI过程域的分类,CMMI软件能力成熟度等级与过程能力等级之间的关系和CMMI过程的可视性等内容。

第3章以"需求管理"过程域为例介绍CMMI的过程域基本概念和组成部件,主要包括需求管理过程域,过程域的内容分类,必要部件,期望部件,参考部件,特定实践,共性实践,过程域的框架结构和成熟度等级中过程域的结构等内容。

第4章介绍CMMI模型的评估方法,主要包括CMMI评估方法简介,选择适合组织商业目标的CMMI模型,剪裁模型,评估类型,选择评估时机,评估定级判断准则,确定评估目标和需求,成立评估组,检查验证客观证据,确定评估结果,报告评估结果,拟定后续改进计划,向评估机构提交报告和归档/销毁关键评估产品等内容。

第5章介绍CMMI模型的能力成熟度等级第2级到第5级的22个过程域的必要部件——目标(包括特定目标和共性目标),期望部件——实践(包括特定实践和共性实践),这些内容是CMMI评估所必须要满足的要求。本章主要包括受管理级、已定义级、定量管理级、持续优化级、能力成熟度的共性目标和共性实践及CMMI-SW过程域间的相互关系等内容。

第6章介绍CMMI模型的能力成熟度等级的2级——受管理级的所有过程域的详细内容,主要包括需求管理,项目策划,项目监督和控制,供方协定管理,测量和分析,过程和产品质量保证和配置管理等过程域。

第7章介绍CMMI模型的能力成熟度等级的3级——已定义级的所有过程域的详细内容，主要包括需求开发，技术解决，产品集成，验证，确认，组织过程聚焦，组织过程定义，组织培训，集成项目管理，风险管理，决策分析和决定等过程域。

第8章介绍CMMI模型的能力成熟度等级的4级——定量管理级的所有过程域的详细内容，主要包括组织过程性能和定量项目管理两个过程域。

第9章介绍CMMI模型的能力成熟度等级的5级——持续优化级的所有过程域的详细内容，主要包括组织革新和部署、原因分析和决定两个过程域。

附录软件生命周期模型对第6.2节中的"软件开发模型"进行了补充。

本书作者曾参与过CMM的认证评估，具有软件过程改进和评估的实践经验。因此，书中内容也结合了作者的一些实践体会，希望能对读者有所帮助。

本书可以作为软件技术人员掌握CMMI的基本知识和核心内容的自学教材，亦可以作为软件组织实施软件过程改进和进行软件能力成熟度评估的指导文献。本书还可以作为软件学院的CMMI课程的教材或参考书。

参加本书编写的人员有罗运模、林经、温四海、李运霞、谢敏、唐宾、刘志方、李华刚、张海宾、陶元庆、王国华。

由于作者经验所限，书中不足或错误之处，敬请读者批评指正。

编　者

2003年7日

目　录

第1章 CMM与CMMI

CMM(Capability Maturity Model for Software)是美国卡耐基梅隆大学软件工程研究所汇集了世界各地软件过程管理者的经验和智慧而产生的软件过程改进的指导性模型。该模型经过世界各地软件组织的实际应用,证明其对软件过程的改进具有建设性的作用。CMM模型取得巨大成功后,人们试图扩展CMM的应用范围,这就促使了CMMI(Capability Maturity Model Integration for Software)模型的产生。

本章介绍CMM与CMMI的相同之处和不同之处,主要包括软件产业的发展历程,CMM的引入,CMMI的提出,CMMI奠定中国软件能力成熟度模型的行业标准和CMM与CMMI的比较等内容。

1.1 软件产业的发展历程

软件产业是随着电子数字计算机(现简称为计算机或电脑)的发明和广泛使用而产生的。它萌芽于20世纪的50年代,20世纪60年代和70年代开始初步形成规模,20世纪80年代起开始获得快速发展。而今,软件产业已逐步从一个弱小的产业部门跃居为新兴的、发展最快的和潜力巨大的产业部门。它代表着一个国家高新技术的水平。没有先进的软件产业,就不可能有先进的信息技术产业;没有先进的信息技术产业,就不可能有先进发达的国民经济。世界上不论是发达国家还是发展中国家都意识到了软件的重要性。

1.1.1 软件的特殊性

通常所说的软件是指计算机软件(简称软件),是相对于计算机硬件而言的。就目前来说,软件包括计算机程序和与之相关的文档。程序控制计算机的运行,而文档则包括程序的操作说明、维护手册等。软件的生产既是人类智慧产品的生产,也是工业产品的生产,因而具有自身的特性。

1. 软件是智力劳动的产物

软件是通过脑力劳动而编写出来的,因而是典型的智力劳动的产物。编写软件与作家写作小说或者写文章有着非常类似的过程。软件中的程序是由一行一行的程序语句(代码)构成的。程序员在编写程序语句时不但要考虑其正确性,而且还要考虑其前后关系及其逻辑性。软件中的文档则一般均用自然语言编写,毫无疑问是智力劳动的产物。这一特性表明,软件产品的生产管理主要是管人。

2. 软件是集体智慧的结果

早期的软件仅包括程序,而且一个程序往往是由一个人来编写,这跟小说的生产是相

似的。这样产生的软件就仅是个体智力劳动的结果。但随着软件产业的形成和发展,软件中的程序的应用范围越来越广,程序的复杂性越来越大,软件的编写就成为集体(几人、几十人、几百人乃至几万人)智力活动的结果。随着程序应用范围的扩大和复杂性的提高,软件文档的数量也迅速增加,因而也需要越来越多的人来编写文档。这一特性表明,软件产品的生产需要工程化的管理方法来进行管理。

3. 软件是难以阅读的作品

一般来说,人们对阅读小说是不会有太大的困难的。软件产品虽然也是人类智力劳动的结果,但它却是难以阅读的。软件中的程序是用计算机语言(不是自然语言)编写成的,并且可以控制计算机的运行,因此阅读起来比较困难。即使是同行(即同是软件工程师),如果没有软件文档的帮助也是难以阅读的。软件中的文档虽然基本上是用自然语言编写的,也是不容易阅读和理解的。这一特性表明,软件产品的生产需要有别于一般的工程化的管理方法来进行管理,也即是需要特殊的、全新的管理方法来管理软件产品的生产。

4. 软件成本高

20 世纪 50 年代,计算机系统应用于非常狭窄的领域,软件的功能单一,规模较小,其成本约占整个计算机系统的 10%～20%。随着计算机技术的进步,生产规模的扩大,计算机硬件的价格不断下降,软件成本在计算机系统中所占的比例越来越大。到 20 世纪 60 年代中期已经增至 50%左右,20 世纪 70 年代以后,软件费用进一步增加。例如,1980 年美国政府的财政年度中,计算机软、硬件与服务费共耗资 570 亿美元,其中软件费用 320 亿美元,占总数的 56%。此后计算机硬件的成本持续降低,而软件成本则不断增加,软件成本在计算机家族总成本所占的比例呈现日益扩大的趋势。如今,在美国软件成本大约已占计算机系统总成本的 90%以上。软件的这一特性,使其已经成为信息产业中至关重要的角色,大有成为信息产业霸主的气势。

5. 软件开发的进度难于控制

计算机软件是一种逻辑系统。设计一个软件系统比设计一个硬件系统所使用的逻辑量要多 10～100 倍。为了完成一个复杂的软件系统,人们常常要考虑建立一个庞大的逻辑体系。此外,同样的软件算法在程序实现上的差别也非常大,加之在软件开发过程中可能遇到各种意想不到的问题,所以投入的资源能否出结果,出什么样的结果,事先很难预料。软件的这一特点,不仅给项目计划和论证工作带来很大的困难,而且使软件开发过程很难保证按预定计划实现。

6. 估计软件工作量很困难

通常,人们执行一项任务时,需要根据其复杂性、工作量及进度要求安排人力,但是软件的工作量是很难估计的。这是因为,一方面软件开发实际上是逻辑思维过程,在写出程序并拿到计算机上运行之前,软件开发的进展情况难以衡量,质量也难于评价,因而其工作量是很难估计的。另一方面,软件规模和复杂性呈指数剧增,开发一个大型软件系统,往往需要成百上千人分工协作。由于软件系统的结构很复杂,各部分联系密切,大量的通信、后勤工作增大了工作量。因此,增加人员,往往不仅不能缩短开发时间,反而会延缓进度,这是与生产一般工业产品的不同之处。软件的这一特性,使其需要一种与一般工业产品生产工作量估计很不同的方法来进行估算。

7. 软件质量难于保证

软件不同于硬件，软件不会用坏，不存在零件更换问题，但不允许存在误差，不能发生错误，否则后果十分严重。医疗系统中的软件错误可能造成生命危险，银行系统中的软件错误会使金融混乱，航空管理系统中的错误会造成飞机失事。例如，美国在一次发射火箭的实验中，由于飞行计划程序里漏掉一个连字符而导致了火箭实验的失败。还有，在美国20世纪60年代所实施的阿波罗登月计划期间，就因为控制登月飞船的程序中少了一个逗号“,”而使飞船发生故障，从而使得该次登月行动失败，而且几乎使宇航员丧失生命。

软件一旦发生错误时，只能在生产现场改正或修改原来的设计。这样，就必须停止生产活动，从而导致重大的经济损失。大型系统在开发时，无法看出是否能正确工作，错误率高，质量很难控制。20世纪90年代，日本投入50亿美元开发第五代计算机的计划，就因为软件无法开发出来而下马。好在这样的损失是由日本政府承担的，如果是一个企业的话，就可能由此而倒闭。

软件的质量问题与其他商品的质量问题有很大不同，因为软件是属于计算机领域的产品，软件设计人员与用户对计算机的了解和想法有很大的距离，程序人员通常以自己的想法去理解用户对软件的要求，而计算机用户对自己所想使用的软件功能和性能在事前也难以说清楚，即使本人是技术人员也不例外，这样在需求分析上就难免存在差距。此外，在软件开发过程中，即使有多种文档，大量的素材仍在程序员的头脑中，软件也只有程序清单，这就使得不了解情况的人很难插上手，最终导致了软件的修改和维护十分困难，有时甚至无法开展。实际上对软件质量最有发言权的是用户，但用户无法也无力参加到软件的质量管理中，这就导致软件设计常有不少随意性，使软件的质量控制成为一个很难解决的问题，以至计算机软件产业普遍存在投入了大额资金和大量人力，而得不到用户满意的产品。如何控制和管理软件产品的质量，是软件行业从一开始就面临的问题，这个问题之所以难以解决在于软件的特殊性。软件的这一特性表明，对其质量的管理需要一种完全不同于一般工业生产管理的方法来保证软件产品的质量。

8. 修正维护软件困难

任何具有一定规模的软件产品都会存在一些错误，即缺陷。实际上即使是正式投入使用的商业软件，尽管采用了各种手段来保证其质量，但也总会存在一定数量的缺陷。美国微软公司的Windows和Office等软件就是典型实例。随着时间的推移，在不同的运行条件下，软件会出现故障，需要维护。但软件的维护与硬件的维护不完全相同。因为软件具有以下的特性。

(1) 软件不是一种实物，而是逻辑元件，软件故障属逻辑故障，不是硬件的“用旧”、“损坏”之类的问题。维护软件不是更换某种备件，而是要纠正逻辑缺陷，使之改正错误，增加适应性，或提高性能。

(2) 当软件系统规模庞大，问题复杂时，经常会发生“纠正一个错误带来更多的新错误”的问题。

(3) 软件修改和扩充表现为改变程序中几条语句或几条指令，当系统投入运行后为适应新增加或变化的设备条件或为增添新功能，经常要求进行维护。因此，软件的维护工作量较大。

(4) 软件产品修正的一个特别的现象是，修改一个缺陷后可能会产生另一个缺陷，这

就是人们常说的“按下葫芦浮起瓢”现象。

由于软件是计算机系统中的逻辑部件而不是物理部件,软件开发是逻辑思维过程,软件的工作量很难估计,进度难以衡量,度量也难以评价,成本高,维护工作繁重。同时软件的复杂度随规模按指数增加,这就需要许多人共同开发一个大型系统。团队开发软件虽然增加了开发力量,但也增加了额外的工作量。组织不严密,管理不善,常常是造成软件开发失败多、费用高的重要原因。软件修正维护困难的特性,使人们面临的不仅是技术问题,更重要的是管理问题。

1.1.2 软件危机

在计算机系统发展的早期(20 世纪 60 年代中期以前),通用硬件相当普遍,软件却是为每个具体应用而专门编写的,这时的软件通常是规模较小的程序,编写者和使用者往往是同一个(或同一组)人。这种个体化的软件环境,使得软件设计通常是在人们头脑中进行的一个隐含的过程,除了程序清单之外,没有其他文档资料保存下来。

从 20 世纪 60 年代中期到 70 年代中期是计算机系统发展的第二代时期,这个时期的一个重要特征是出现了“软件作坊”,当时的微软公司就是这种软件生产方式的一个代表。此时,尽管软件的生产是作坊式生产,但还是生产了一些很有代表性的产品,使得产品软件的使用广泛起来。然而,“软件作坊”基本上仍然沿用早期形成的个体化软件开发方法。随着计算机应用的日益普及,软件数量急剧膨胀。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。上述种种软件维护工作,以令人吃惊的比例耗费资源。更严重的是,许多程序的个体化特性使得它们最终成为不可维护的。“软件危机”就这样开始出现了!

为什么会出现“软件危机”呢?

软件不同于一般程序,它是由许多实现各自功能的程序组成,它的一个显著特点是规模庞大。例如,美国第四代宇宙飞船的软件规模呈指数增长,20 世纪 70 年代末穿梭号宇宙飞船的软件包含 4 000 万行目标代码。假设一个人一年可以开发出一个一万行的程序,为了开发一个 4 000 万行的软件,是否集中 4 000 人的力量一年就可以完成呢?绝对做不到!因为代码长度增加了 4 000 倍,程序复杂程度的增加远远超过 4 000 倍。而且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,例如,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。

软件本身独有的特点确实给开发和维护带来一些客观困难,但是人们在开发和使用计算机系统的长期实践中,也确实积累和总结出了许多成功的经验。如果坚持不懈地使用经过实践考验证明是正确的方法,许多困难是完全可以克服的,过去也确实有一些成功的范例。但是,目前相当多的软件专业人员对软件开发和维护还有不少糊涂观念,在实践过程中或多或少地采用了错误的方法和技术,这可能是导致软件问题发展成软件危机的主要原因。

与软件开发和维护有关的许多错误认识和作法的形成,可以归因于在计算机系统发展的早期软件开发的个体化特点。错误认识和作法主要表现为忽视软件需求分析的重要性,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。

事实上,对用户要求没有完整准确的认识就匆忙着手编写程序是许多软件开发工程失败的主要原因之一。只有用户才真正了解他们自己的需要,但是许多用户在开始时并不能准确具体地叙述他们的需要,软件开发人员需要做大量深入细致的调查研究工作,反复多次地与用户交流信息,才能真正全面、准确、具体地了解用户的要求。对问题和目标的正确认识是解决任何问题的前提和出发点,软件开发同样也不例外。急于求成,仓促上阵,对用户要求没有正确认识就匆忙着手编写程序,这就如同不打好地基就盖高楼一样,最终必然垮台。

一个软件从定义、开发、使用和维护,直到最终被废弃,要经历一个漫长的时期,这就如同一个人要经过胎儿、儿童、青年、中年、老年,直到最终死亡的漫长时期一样。通常把软件经历的这个漫长的时期称为生命周期。软件开发最初的工作应是问题定义,也就是确定要求解决的问题是什么;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;接下来应该进行需求分析,也就是深入具体地了解用户的要求,在所要开发的系统(不妨称之为目标系统)必须做什么这个问题上和用户取得完全一致的看法。经过上述软件定义时期的准备工作才能进入开发时期,而在开发时期首先需要对软件进行设计(通常又分为总体设计和详细设计两个阶段),然后才能进入编写程序的阶段,程序编写完之后还必须经过大量的测试工作(需要的工作量通常占软件开发全部工作量的40%～50%)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%。

另一方面还必须认识到程序只是完整的软件产品的一个组成部分,在上述软件生命周期的每个阶段都要得出最终产品的一个或几个组成部分(这些组成部分通常以文档资料的形式存在)。软件专家曾经指出:"软件是程序以及开发、使用和维护程序需要的所有文档。"这也就是对软件的定义。所以,一个软件产品必须由一个完整的配置组成,应该清除只重视程序而忽视软件配置其余成分的糊涂观念。

做好软件定义时期的工作,是降低软件成本提高软件质量的关键。如果软件开发人员在定义时期没有正确全面地理解用户需求,直到测试阶段或软件交付使用后才发现"已完成的"软件不完全符合用户的需要,这时再修改就为时已晚了。

严重的问题是,在软件开发的不同阶段进行修改需要付出的代价是很不相同的,在早期引入变动,涉及的面较少,因而代价也比较低;而在开发的中期软件配置的许多成分已经完成,引入一个变动要对所有已完成的配置成分都做相应的修改,不仅工作量大,而且逻辑上也更复杂,因此付出的代价剧增;在软件"已经完成"时再引入变动,当然需要付出更高得多的代价。根据美国一些软件公司统计资料,在后期引入一个变动比在早期进入相同变动所需付出的代价高2～3个数量级。

通过上面的论述不难认识到,轻视维护是一个最大的错误。许多软件产品的使用寿命长达10年甚至20年,在这样漫长的时期中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(如硬件或系统软件更新换代)还必须相应地修改软件以适应新的环境,特别是必须经常改进或扩充原来的软件以满足用户不断变化的需要。所有这些改动都属于维护工作,而且是在软件已经完成之后进行的,因此维护是极端艰巨复杂的工作,需要花费很大代价;统计数据表明,实际上用于软件维护的费用占软件总费用的55%～70%。

1.1.3 软件工程

软件开发不是某种个体劳动的神秘技巧,而应该是一种组织良好、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人类长期以来从事各种工程项目所积累的行之有效的原理、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。

应该推广使用在实践中总结出来的开发软件的成功的技术和方法,并且研究探索更好更有效的技术和方法,尽快消除在计算机系统早期发展阶段形成的一些错误观念和做法。

应该开发和使用更好的软件工具。正如机械工具可以"放大"人类的体力一样,软件工具可以"放大"人类的智力。在软件开发的每个阶段都有许多繁琐重复的工作需要做,在适当的软件工具辅助下,开发人员可以把这类工作做得既快又好。如果把各个阶段使用的软件工具有机地集合成一个整体,支持软件开发的全过程,则称为软件工程支撑环境。

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软件工程,它是指导计算机软件开发和维护的工程学科。采用工程的概念、原理、技术和技术方法结合起来,这就是软件工程。

自从1968年在联邦德国召开的国际会议上正式提出并使用了"软件工程"这个术语以来,研究软件工程的专家学者们陆续提出了100多条关于软件工程的准则或"信条"。著名的软件工程专家波汉姆(Boehm)综合这些学者们的意见,并总结了TRW公司多年开发软件的经验,于1983年在一篇论文中提出了软件工程的7条基本原理。他认为这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。这7条原理是互相独立的,其中任意6条原理的组合都不能代替另一条原理,因此,它们是缺一不可的最小集合,然而这7条原理又是相当完备的,人们虽然不能用数学方法严格证明它们是一个完备的集合,但是,事实证明在此之前已经提出的100多条软件工程原理都可以由这7条原理的任意组合蕴含或派生。

软件工程的7条基本原理简要内容如下:

1. 使用分阶段的生命周期计划严格管理

有人经统计发现,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1条基本原理是吸取了前人的教训而提出来的。

在软件开发与维护的漫长的生命周期中,需要完成许多性质各异的工作。这条基本原理意味着,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。波汉姆(Boehm)认为,在软件的整个生命周期中应该制定并严格执行6类计划,它们是项目概要计划,里程碑计划,项目控制计划,产品控制计划,验证计划,运行维护计划。

不同层次的管理人员都必须严格按照计划各尽其职地管理软件开发与维护工作,绝不能受客户或上级人员的影响而擅自背离预定计划。