

# CMM/CMMI 与软件生命周期模型关系的研究\*

邢彬彬, 姚 郑

(中国科学院 研究生院 计算与通信工程学院, 北京 100049)

**摘 要:** 对 CMM/CMMI 和主流软件生命周期模型的关系进行了分析, 认为: a) CMM/CMMI 与软件生命周期模型都是软件活动过程化的产物。在软件过程化进程中, CMM/CMMI 是软件过程化成熟期的成果; 软件生命周期模型则是软件过程化前期的成果。b) 由于发展背景不同, 需要处理的问题各异, 使得软件生命周期模型关注于工程活动; CMM/CMMI 关注于软件开发活动的整体。c) 尽管 CMM/CMMI 编写者的初衷是保持 CMM/CMMI 与特定软件生命周期模型的无关性, 但是其产生的时代背景却使 CMM/CMMI 受到了软件生命周期模型的影响。

**关键词:** 软件生命周期模型; 软件能力成熟度模型; 能力成熟度模型集成

中图分类号: TP311.5 文献标志码: A 文章编号: 1001-3695(2007)11-0065-05

## Study of relation between CMM/CMMI and software life cycle models

XING Bin-bin, YAO Zheng

(College of Computing & Communication Engineering, Graduate School, Chinese Academy of Sciences, Beijing 100049, China)

**Abstract:** Based on the analysis of CMM/CMMI and major software life cycle models, this paper put forward the following points: a) CMM/CMMI and software life cycle model were both results of the processization of software activities and they both focused on the processization of software activities. To be specific, CMM/CMMI was the result of the maturity stage, and software life cycle model was the result of the early stage. b) Owing to different development backgrounds and different problems to resolve, software life cycle model focused on engineering activities, while CMM/CMMI focused on non-engineering activities. c) Though the CMM/CMMI authors had tried to make CMM/CMMI be independent of any certain software life cycle model, the originating era of CMM/CMMI did leave CMM/CMMI with the brand of software life cycle models.

**Key words:** software life cycle model; CMM; CMMI

目前, 能力成熟度模型(SW-CMM/CMMI)和各种软件生命周期模型在软件产业都得到了广泛使用, 而且经常被同时使用。当一个组织在运用 CMM/CMMI 进行软件过程改进的过程中, 往往会被其中涉及到的各种软件生命周期模型及其与 CMM/CMMI 的关系所困惑, 从而影响了过程改进的效果。

对 CMM/CMMI 与软件生命周期模型之间整体关系的认识, 是软件组织对软件开发活动进行宏观决策的依据, 将决定软件开发活动的方向。例如, 对于整个组织的软件开发活动, 软件生命周期模型用于解决哪些问题, CMM/CMMI 用于解决哪些问题, 在应用中可能产生什么结果? 对于不同类型的项目, 这两者的应用是否存在不同? 这些宏观的决策在组织中起着非常重要的作用。为此, 本文主要讨论了 CMM/CMMI 与软件生命周期模型之间的关系: a) 在发展历史上, 两者之间的关系; b) 在软件工程领域, 两者关注范围的比较; c) 两者内容的相互影响。

通过以上三方面的整体关系讨论, 将有助于进一步分析两者具体内容之间的内在关联, 加深对两者的理解和认识。

## 1 发展历史上的关系

### 1.1 在软件行业发展历史上的地位

软件行业迄今已有近五十年的发展历程, R. Lai 认为其中

有三次较明显的发展浪潮<sup>[1]</sup>: 第一次浪潮以瀑布式生命周期和结构化方法为特征; 第二次浪潮是过程成熟度运动; 第三次浪潮将是预期中的软件工业化。

第一次浪潮是对之前没有结构的软件活动的改进。通过瀑布模型和结构化分析设计方法, 人们进一步理解了软件开发活动。第二次浪潮改进的范围更广, 覆盖了软件开发过程及其支持活动的各个方面。作为第二次浪潮的 CMM 模型, 很可能会促使软件工业在工程化的道路上更进一步, 使软件工业成为以过程为中心的工业, 类似于制造业。

R. Lai 对第一次浪潮和第二次浪潮的划分, 恰如其分地表明了软件生命周期模型与 CMM 在软件行业发展过程中的地位。

软件生命周期模型的发展始自 W. Royce 在 1970 年发表的瀑布模型。瀑布模型将软件开发活动分为需求分析、设计、编码、测试等几个过程。这样的划分具有里程碑意义, 将人们从之前的没有结构的软件活动中解脱出来, 开始从更高的角度审视软件开发活动。

在瀑布模型的基础上, 各种软件生命周期模型不断被提出。后续的各种软件生命周期模型是在与瀑布模型相同的层次上对各种软件开发活动进行描述和总结。在这一时期, 软件生命周期模型是软件工程领域的重要研究方向, 各种软件生命

周期模型总结了产业界的各种软件开发活动的特点,加深了对于软件开发活动的认识。

过程成熟度运动始于 W. Humphrey 在 1986 年的工作,他在软件过程中采用统计质量控制原理,并将 Crosby 的成熟度量化的思想引入,形成成熟度等级的概念。在 W. Humphrey 的工作基础上,软件工程研究所先后发表了能力成熟度框架<sup>[2]</sup>、成熟度问卷<sup>[3]</sup>、CMM1.0、CMM1.1、CMMI1.0 和 CMMI1.1。

这些工作对软件行业产生了巨大影响,在世界范围内引起了软件从业者的关注。各种其他学科的成熟度模型不断被提出,其他标准组织也不断接受过程思想,在其标准中加入过程元素。例如国际标准化组织广泛应用的标准 ISO9000,在其最新版本 2000 版中,也加入了过程思想<sup>[4]</sup>,在其引言中指出“本标准鼓励在建立、实施质量管理体系以及改进其有效性时采用过程方法,通过满足顾客要求,增强顾客满意”“过程方法的优点是对诸过程的系统中单个过程之间的联系以及过程的组合和相互作用进行连续的控制”。很多国家和组织也都开始了对软件过程的研究并形成了自己的模型和标准<sup>[5]</sup>。

综上所述,软件生命周期模型和 CMM/CMMI 在软件行业发展史上分别代表了第一次浪潮和第二次浪潮,影响都是深远和巨大的。

## 1.2 在软件过程化进程中的位置

W. Humphrey 第一次在软件产业中提出过程的思想。但笔者认为,软件产业的过程化并不是在 W. Humphrey 明确提出过程概念后才开始的。实际上,软件产业的过程化可以追溯到瀑布模型和各种软件生命周期模型<sup>[6]</sup>。

在瀑布模型产生之前,软件开发以编码活动为主,没有过程和结构,不确定和混乱是软件开发的重要特点。软件已开始大量用于商业应用中,软件的规模也越来越大,软件开发的理论成为业界的研究对象。

结构化分析设计方法的提出使得软件开发活动逐渐有了划分,不再局限于编码活动。该方法的分步骤进行及各步骤间相对独立的特点为瀑布模型的产生提供了技术可行性。

瀑布模型正是在这样的背景下产生的。瀑布模型与结构化分析设计方法一起对软件开发活动进行了最初的过程化:将没有结构的软件开发活动划分为需求分析、设计、编码、测试等几个过程。与瀑布模型类似,其他的软件生命周期模型也是对软件开发活动进行着各种过程化,从不同的角度识别过程,确定过程之间的相互作用,从而更好地对软件开发活动进行描述和总结。可见,软件生命周期模型是软件过程化进程中的早期成果。在技术相关的活动被过程化之后,其他方面的问题逐渐表现出来,如项目管理、风险问题。

在一些使用瀑布模型比较成熟的软件组织,技术活动相对比较成熟,管理活动和其他活动成为进一步的改进方向,这些非技术活动包括项目管理活动、软件开发相关的支持活动(如需求管理和配置管理)、提高软件组织生产能力的活动(如统计质量控制原理的应用)等。在这些软件组织,非技术活动的过程化得到了相当大的发展,相关理论和实践经验由 W. Humphrey 带到了 SEI,并据此形成了 CMM/CMMI 的早期版本。从此 CMM/CMMI 被不断使用,过程思想逐渐形成了巨大的影

响。可见,CMM/CMMI 模型是软件过程化进程中的后期产物。

## 2 两者关注点的异同

在软件工程领域,软件生命周期模型更关注于软件开发工程的过程,在这个过程中如何对各项活动进行工程化;CMM/CMMI 则比较关注软件开发活动的整体,如何对软件开发活动的各个方面进行过程化,其中包括工程活动的过程化。

### 2.1 软件开发活动的分类

软件开发活动是一个组织中与软件开发相关活动的统称。软件开发中包含的活动很多,为方便讨论,这里使用工程活动指代工程类的活动,用非工程活动指代除工程类活动以外的其他活动。

工程活动是与软件建造本身密切相关的活动,包括确定软件要解决的实际问题、构造软件以解决该问题、对构造的软件进行维护等。例如需求分析、设计、编码、测试等活动。正是这些活动建造出最终软件,因此是软件开发活动中的核心活动。

非工程活动是为了使工程活动能进行得更加顺利、更有效率而对工程活动进行的支持和保障,如项目管理活动、配置管理活动等。这些活动针对的对象是工程活动,不能脱离工程活动而单独存在。这些非工程活动对软件构造所起的作用是通过工程活动产生的,因此是工程活动的支持和保障。

区分工程活动与其他活动的原因在于,在软件开发相关活动中,工程活动与其他活动的位置不同。工程活动在软件开发相关活动中处于核心位置,非工程活动处于支持和保障位置。

### 2.2 软件生命周期模型关注于工程活动

在软件生命周期模型的领域内有若干种模型,这些模型的主要关注点在工程活动,部分模型也关注一些非工程活动。

瀑布模型将软件开发活动分为需求分析、设计、编码、测试等几个阶段。这几个阶段是对工程活动的划分。瀑布模型没有再涉及其他方面的活动。因此瀑布模型关注于工程活动。

原型模型则主要是为了解决需求获取的难题而创建原型用于需求的获取和确认,再将需求转换为软件系统,其主要内容集中在软件开发本身。因此原型模型也关注于工程活动。

螺旋模型的主要贡献在于明确提出迭代概念和风险的问题,并指出在项目定义、需求、设计等阶段均存在风险,需要重点考虑,并通过多次迭代的原型主动诱发风险。风险管理不属于工程活动的范围,但笔者仍然认为螺旋模型的主要内容是工程方面的。因为对于非工程活动,螺旋模型仅考虑了风险问题,而风险管理仅是非工程活动的一个小部分,不足以依此推断螺旋模型主要关注于非工程活动。

增量模型将瀑布模型的顺序化与多次迭代相结合。每个增量开发都是一次瀑布模型的过程,强调每一个增量均发布一个可运行版本,以满足客户和市场需要。增量模型主要考虑当需要快速推出可运行版本,而该版本不需要完整的功能时,在工程活动上的解决方案,因此增量模型也关注于工程活动。

图 1 描述了几种软件生命周期模型的关注点。可以看出,软件生命周期模型关注于工程活动。

### 2.3 CMM/CMMI 关注于软件开发活动的整体

CMM/CMMI 的关注点是软件开发活动的整体。将整个软

件开发活动划分成了不同的过程域,再针对各个过程域分别进一步描述,从而将软件开发活动整体进行了过程化。在此基础上,CMM/CMMI 给出了对软件开发活动进行不断改进的方法——统计过程控制方法。

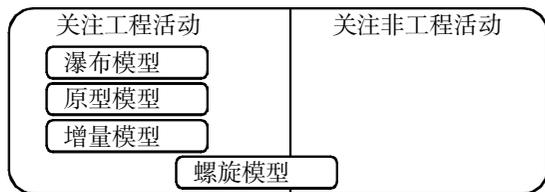


图 1 几种软件生命周期模型的关注点

CMM对关键过程域的分类如表 1 所示。

表 1 CMM中关键过程域的分类

等级	分类		
	管理	组织	工程
第五级优化级		技术变更管理	
第四级已管理级	定量过程管理	过程变更管理	缺陷预防
第三级已定义级	集成软件管理 组间协调	组织过程定义 组织过程焦点 培训大纲	软件产品工程 同行评审
第二级 可重复级	需求管理 软件项目策划 软件项目跟踪与 监督		
第一级初始级	软件子合同管理 软件质量保证 软件配置管理		

CMM将过程域分为管理类、组织类、工程类三类。

对于管理类的关键过程域,CMM 有比较完整而清晰的活动划分和改进目标。从第二级项目级的项目策划和项目跟踪等活动,进一步到第三级组织级的按照已定义的软件过程进行项目管理,再到第四级的定量管理。

对于组织类的关键过程域,CMM 也提供了比较清晰的改进方向和方案。从第三级开始,在项目级别的基础上,组织开始对软件过程进行管理和定义,形成组织标准过程;在第四级,对过程进行量化预测和控制;到第五级对过程不断进行改进。

CMM中工程类的关键过程域,从第三级开始包含软件产品工程,并要求进行同行评审;到第四级对质量活动的明确要求;再到第五级的缺陷预防和技术变更管理。

可以看出,CMM模型的内容比较全面,覆盖了软件开发活动的各个方面。工程类过程域是其中的一个方面,并不是其关注的焦点,地位与其他方面的过程域相同。CMMI 模型的情况与 CMM类似,并且进一步加强了工程类过程域。这里不再赘述。

#### 2.4 两者关注点不同的原因分析

软件生命周期模型关注于工程活动;CMM/CMMI 关注于软件开发活动的整体。这与工程活动和非工程活动在软件开发活动中的位置不同有关系,也与软件生命周期模型和 CMM/CMMI 发展的背景有关。

如前所述,在软件开发活动中,工程活动承担着从需求到软件的转换过程。这一过程是软件开发活动的核心活动。有了这些活动,软件的产出才成为可能,而其他非工程活动是为了让工程活动进行得更顺利、更高效而采取的辅助和保障活动。因此,在软件过程化进程中,作为核心活动的工程活动会

首先被过程化,然后非工程活动才会被过程化。而软件生命周期模型正是工程活动被过程化的成果;CMM/CMMI 则是将各类活动过程化的成果。

在瀑布模型产生之前,软件开发以程序编制活动为主,没有需求分析、设计、测试等活动。这些活动与程序编制活动没有结构地混合在一起,组成软件构造的工程活动。这样的工程活动难以预测和控制,使得管理活动难以开展。此时首先需要对工程活动进行过程化,只有当处于核心位置的工程活动能够有序地进行,非工程活动才能根据工程活动的特点开展。

结构化分析设计方法对工程活动进行了改进,为从需求转化到软件的过程提供了完整的转换方法,使得工程活动能够按步骤、有次序地进行。瀑布模型是在结构化分析设计方法的基础上对软件开发活动进一步的过程化。

在瀑布模型的基础上,其他软件生命周期模型希望能够更好地对软件开发活动进行描述和总结,于是从不同的角度,对于不同类型的软件开发过程提出了若干种软件生命周期模型。由于这些模型的开发是在瀑布模型的背景下进行的,其研究范围与瀑布模型接近,关注于工程活动较多。

工程活动被过程化之后,非工程活动的过程化也被重视起来。与此同时,美国国防部对软件承包商提出在进度和成本预算内开发出高质量软件的要求,促使软件产业界对相关问题进行研究。美国国防部的一个研究小组分析软件危机后得出结论:“军用软件开发中,当前最主要的问题不是技术问题,而是管理问题<sup>[8]</sup>”。也就是说,不是工程活动的问题(技术问题),而是非工程活动的问题(管理问题)。

SEI 在 Watts Humphrey 的领导下,收集产业经验并借鉴传统行业的质量理论,在此基础上进一步发展并形成了 CMM/CMMI 模型。由于这样的发展背景,CMM/CMMI 关注于包括工程活动在内的软件开发活动的各个方面。

### 3 软件生命周期模型对 CMM/CMMI 的影响

#### 3.1 CMM/CMMI 的产生背景

从前面的讨论可以看出,在软件生命周期模型被提出的过程中,CMM/CMMI 基本没有影响软件生命周期模型的发展,在各种软件生命周期模型中都没有受到 CMM/CMMI 影响的内容。但是,在 CMM/CMMI 模型中,却可以看到瀑布模型的影子。本节将讨论瀑布模型在整体上对 CMM/CMMI 的影响。

CMM/CMMI 产生的一个目的是为了美国政府和美国国防部能有效地评估软件承包商。而美国政府和美国国防部都有自己的软件开发标准。从 CMM/CMMI 与美国国防部的软件开发标准之间的关系,可以推断出一些结论。

1988 年之前,美国国防部的软件研发标准是 DOD—STD—2167。该标准将瀑布模型作为美国国防部军用软件的开发规范<sup>[9]</sup>。1988 年,重新修订 DOD—STD—2167,发布了 DOD—STD—2167A。它允许但未制定新的模型如螺旋模型等。Bary Boehm 报告指出:不幸的是,2167A 所参考的军标 MILSPECS 和说明性的例子依然是面向瀑布模型的,因此依然继续使用瀑布模型<sup>[9]</sup>。1994 年,美国国防部发布了 MIL—STD—498 《军用标准——软件开发和文档》,以此取代了上述

DOD—STD—2167A 等三部军用标准。在与软件生命周期模型的关系上, MIL—STD—498 标准提高了与增量模型和演化模型的兼容性<sup>[10]</sup>。1995 年 ISO/IEC 颁布了关于软件研发工业上贯彻的国际标准, 即 ISO/IEC12207—1995 《信息技术——软件生命周期过程》。IEEE 根据 ISO/IEC12207 的内容制定了标准 IEEE/EIA12207.0。1997 年 IEEE/EIA 协会又相继颁布了两部指南, 12207.1 和 12207.2。1998 年 4 月, 美国国防部批准采用以上三个标准取代 MIL—STD—498 军用标准。ISO/IEC12207 提供了一个软件生命周期过程的高层次的通用框架。在此框架下可以容纳各种软件生命周期模型, 并给出了裁减指南<sup>[11]</sup>。

由此可以看出, 在 1988—1998 年的十年间, 美国国防部的软件开发标准所允许采用的软件生命周期模型从瀑布模型扩展到了各种软件生命周期模型。可以判断在 1988 年之前, 美国国防部的软件开发标准所允许采用的软件生命周期模型是瀑布模型, 承接美国国防部软件项目的软件企业应该会按照瀑布模型进行软件开发。

CMM 模型的 1.0 版本于 1991 年提出, 最早期的版本于 1987 年提出。因此 CMM 主要内容的确定是在 20 个世纪 80 年代, 其主要实践经验都来源于当时承接美国国防部项目的软件企业的软件开发活动。并且, 对于非美国国防部承包商的其他软件企业, 也在广泛使用瀑布模型, “在 1985 年, 几乎所有明确定义了软件开发生命周期的项目都采用了瀑布模型”<sup>[12]</sup>。由此可以推断, CMM 的主要实践经验都来源于瀑布模型的实践经验。

上文提到, 到 1998 年, 美国国防部的标准所允许采用的软件生命周期模型从瀑布模型扩展到了各种软件生命周期模型, 这是美国国防部对非瀑布生命周期模型的认可, 也反映了非瀑布生命周期模型在产业界的应用已经较为普遍。

CMMI 模型的第一个版本是在 1999 年提出的。该模型的产生原因之一是对 CMM 模型进行改进, 处理 CMM 模型在使用中发现的问题, 吸收产业界新的实践经验。非瀑布生命周期模型的普遍应用所产生的实践经验, 也对 CMMI 模型产生了一些影响。

### 3.2 瀑布模型对 CMM 的影响

CMM 作为通用的模型, 要适用于各种工程环境中, 因此既不提倡也不反对各种软件生命周期模型的使用。在 CMM 中明确对软件生命周期模型的问题进行了说明<sup>[7]</sup>: “关键实践并不意味着限制软件生命周期的选择。已多次使用某一个特定软件生命周期的人可能会在关键实践的组织和结构中察觉到该生命周期的元素。可是, 并不存在鼓励或阻碍使用任何特定软件生命周期的企图。术语‘阶段’用来指示软件项目工作的一种已定义的划分, 但它并不与任何特定的软件生命周期相联系。正如在关键实践中所用的那样, 阶段可以指严格的序列阶段, 也可以指部分重叠的和迭代的阶段。”

上述说明基本上表明了 CMM 编写者的初衷: 避免 CMM 受到软件生命周期模型的影响; 希望 CMM 模型能够适用于各种软件生命周期模型。尽管 CMM 的编写者存在这样的愿望, 但 CMM 产生的时代背景使 CMM 受到了瀑布模型的影响。瀑

布模型对于 CMM 的主要影响表现在: CMM 将统计质量控制理论应用在软件行业; CMM 过于强调需求的固化和跟踪; CMM 过于强调瀑布模型的产出物和典型活动。

以上几点通常会使得实施 CMM 的软件组织认为与 CMM 最匹配的软件生命周期模型是瀑布模型。下面将分别对这几点进行讨论。

1) 将统计质量控制理论应用在软件行业 统计质量控制理论主要应用于传统行业中的组织。这些组织的生产特点是产品的规模化生产, 生产工艺稳定, 同一生产流程多次重复, 因此可以对某一生产流程的数据进行统计。通过与统计结果的比较确定该次生产流程的质量, 从而对产品质量进行控制。而软件行业的特点是难以规模化生产。规模化生产是软件工程学科的发展目标。本文认为, 在瀑布模型成熟应用的年代, 在某些适合应用瀑布模型软件开发领域, 已经存在类似项目不断重复、生产流程不断重复的情形。这种情形还不是规模化生产, 但已经有了规模化生产的征兆, 如流程的稳定重复。瀑布模型简单而易于理解, 在适合于使用瀑布模型的软件开发活动当中, 各项活动的稳定性和可重复性较强。在瀑布模型盛行的 20 世纪 70 ~80 年代这二十年间, 大厂商和美国国防部的软件合同商都在广泛使用瀑布模型。由于瀑布模型和结构化分析设计方法的成熟应用, 软件的需求分析、设计、编码、测试活动相对比较成熟可控, 不同项目之间的可比性较强, 数据有较强的统计意义, 可以应用统计质量控制理论对各类活动进行统计和质量控制。

2) CMM 过于强调需求的固化和跟踪 需求的开发和管理, 由于其对于软件开发的重要性和复杂性, 一直是各种软件生命周期模型关注的问题。原型模型通过原型来诱导和确认需求; 增量模型通过每次增量发布的可运行的版本来确认需求; 瀑布模型则通过需求规格说明书来确认需求。

瀑布模型从需求开始, 以需求规格说明书来固化需求, 强调后续活动与需求的一致性。瀑布模型的这一特点与 CMM 模型的需求管理关键过程域中的部分内容非常类似。需求管理关键过程域规定要建立需求基线作为后续工作的基础, 要对需求进行跟踪。相对于原型模型通过原型对需求不断的精化, 增量模型通过可运行的版本对需求进行确认。CMM 中需求管理的思想更接近于瀑布模型。

3) CMM 过于强调瀑布模型的产出物和典型活动 CMM 模型中软件产品工程关键过程域中, 规定了需要进行的工程活动。CMM 规定的工程活动包括需求分析、体系结构设计、详细设计、编码、集成测试、系统测试和验收测试等。CMM 并没有说明这些工程活动要严格按照瀑布模型的顺序进行, 也没有说明这些活动不能重叠和迭代。但是相比较其他软件生命周期模型的产出物和典型活动, 这些活动与瀑布模型过于相似。

### 3.3 软件生命周期模型对 CMMI 的影响

从软件行业的角度看, CMMI 吸收了最近若干年软件产业的一些实践经验, 也参考了 CMM 在使用过程中的反馈问题, 因此内容上受到了新的生命周期模型的影响。这些影响主要表现在两个方面:

a) 修正 CMM 中受到瀑布模型影响的内容。瀑布模型对 CMM 的影响在 3.2 节已有描述。其中 CMM 过于强调需求的固化和跟踪以及 CMM 过于强调瀑布模型的产出物和典型活动在 CMMI 中都得到了修正。CMMI 中的需求管理过程域不再要求建立需求基线;在相关工程类过程域中也不再强调瀑布模型的产出物和典型活动,而代之以各种类型的产出物和典型活动。这一修正也从侧面支持了本文 3.2 节中的部分观点。但对于将统计质量控制理论应用在软件行业,CMMI 模型没有变化,依然以统计质量控制理论作为过程改进的主导思想。

b) 吸收了迭代式生命周期模型的思想。迭代式生命周期模型典型的如螺旋模型、增量模型、统一软件开发模型等,在产业界已经普遍应用。通过迭代来规避风险、确认需求等优秀实践是产业界比较新的实践成果。CMMI 模型吸收了迭代的思想,认为项目过程本身是一个不断演化、经过多次迭代的过程,在模型中多处反映了这一思想。

例如 CMMI 原文中对需求管理 SP1.2 的解释<sup>[13]</sup>:“Requirements evolve throughout the project, especially as described by the specific practices of the Requirements Development process area and the Technical Solution process area. As the requirements evolve, this specific practice ensures that project participants commit to the current, approved requirements and the resulting changes in project plans, activities, and work products.”可翻译为:“在项目过程中,需求不断进化,特别是在需求开发和技术解决方案过程域的特定实践中。在需求不断进化的情况下,本特定实践确保项目的参与人对当前的、经过批准的需求达成一致,并对后续的项目计划、活动和工作产品的变更达成一致。”

又如在对产品集成过程域的介绍性说明中提到:“Product integration is more than just a one-time assembly of the product components at the conclusion of design and fabrication. Product integration can be conducted incrementally, using an iterative process of assembling product components, evaluating them, and then assembling more product components. This process may begin with analysis and simulations (e. g., threads, rapid prototypes, virtual prototypes, and physical prototypes) and steadily progress through increasingly more realistic incremental functionality until the final product is achieved.”可翻译为:“产品集成不只是在设计和建造结束时对产品组件的一次性组装。产品集成通常增量地进行,采取集成产品组件、评价它们、再集成更多的产品组件这样一个迭代的过程。这一过程可以开始于分析和模拟(如相关串联、快速原型、虚拟原型和实体原型),并稳步进展,逐渐增加更多实现了的功能,直至达成最终产品。”

这些内容已经完全符合迭代式生命周期模型的思想。类似以上的内容,在 CMMI 模型中还有很多,本文不再一一赘述。

CMMI 对于迭代思想的吸收使之更加贴近各种迭代式生命周期模型,使得使用这些生命周期模型的项目能更加顺畅地

符合 CMMI 的要求。

#### 4 结束语

本文从整体上讨论了 CMM/CMMI 与软件生命周期模型的关系。讨论了两者在发展历史中的关系、两者关注范围的比较以及两者内容的相互影响。为进一步研究两者内容上的具体联系和相互影响提供了基础,对 CMM/CMMI 的实施活动也具备一定的参考意义。

#### 参考文献:

- [1] AI R. The move to mature process[ J]. IEEE Software, 1993, 10 (4): 14-17.
  - [2] HUMPHREY W S. Characterizing the software process: a maturity framework[ R]. Pittsburgh: Carnegie Mellon Software Engineering Institute, 1987.
  - [3] HUMPHREY W S, SWEET W L. A method for assessing the software engineering capability of contractors[ R]. Pittsburgh: Carnegie Mellon Software Engineering Institute, 1987.
  - [4] International Standards Organization. ISO 9001—2000, Quality management systems-requirements[ S]. 2000.
  - [5] ZAHARAN S. 软件过程改进[ M]. 陈新,等译. 北京:机械工业出版社, 2002: 46-47.
  - [6] FUGGETTA A. Software process: a roadmap[ C] // FINKELSTEIN A. Future of software engineering. Proc of the 22nd International Conference on Software Engineering (ICSE2000). New York: ACM Press, 2000: 25-34.
  - [7] PAULK M C, et al. Key practices of the capability maturity model SM, version 1.1[ R]. Pittsburgh: Carnegie Mellon Software Engineering Institute, 1993.
  - [8] Department of Defense. Report of the defense science board task force on military software[ R]. Washington, D C: Office of the Under Secretary of Defense for Acquisition, 1987.
  - [9] BROOKS F P. 人月神话[ M]. 汪颖,等译. 北京:清华大学出版社, 2002: 367.
  - [10] US Department of Defense. MIL-STD-498, Military standard-software development and documentation[ S]. 1994.
  - [11] ISO/IEC 12207—1995. Information technology software life cycle processes[ S]. 1995.
  - [12] POLLICE G. How far have we come? [ EB/OL]. [ 2006-09-25]. <http://www-128.ibm.com/developerworks/rational/library/dec04/pollice/index.html>.
  - [13] CMMI Product Team. Capability maturity model integration( CMMI) version 1.1 ( CMMI-SE/SW/ IPPD, v1.1) staged representation[ R]. Pittsburgh: Carnegie Mellon Software Engineering Institute, 2001: 86-87.
- 
- (上接第 42 页)
- [9] 王占权,云庆夏,杨东援.改进的遗传规划研究[ J].系统工程与理论实践,2000(5):66-67.
  - [10] 王占权,云庆夏.工作面设备选型的遗传规划[ J].中国铝业,1999,23(5).
  - [11] ZHANG Meng-jie, CIESIELKI V. Genetic programming for multiple class object detection[ C] // FOO N. Proc of the 12th Australian Joint Conference on Artificial Intelligence. Berlin: Springer-Verlag, 1999.
  - [12] 查志琴,高波,郑成僧.遗传编程实现的研究[ J].计算机应用,2003,23(7):137-139.
  - [13] 何登旭.遗传规划设计与应用[ J].广西民族学院学报:自然科学版,1999,5(3):31-33.