一种基于CMM的软件过程模型*)

A Software Process Model Based on CMM

叶书元 周 鹏 顾 庆 陈道蓄

(南京大学计算机软件新技术国家重点实验室 南京210093)

Abstract CMM is a mature model for process assessment and improvement. But it doesn't include exercisable methods to achieve process improvement. So, some institutes put forward a number of CMM-based process improvement models which are of operable. V-Model is a widely used process model in German and Europe, it mainly focuses on project process modeling and it puts forward many operable methods for process improvement. Here we discuss how V-Model realizes operable project process improvement, and put forward an CMM-based organizational software process model COPM. Later we discuss how to realize this model on our platform CPMS (CMM-based software Process Management & control System).

Keywords CMM, V-Model, Process modeling, CMM key process area

1 引言

CMM(Capability Maturity Model,能力成熟度模型)是从80年代中期开始,由美国国防部资助,卡耐基·梅隆大学(CMU)的软件工程研究所(SEI)研究提出的描述有效的软件过程单元的框架。SEI提出的这一成果得到了众多国家软件产业界的认可,并且在北美、欧洲和日本等国家得到了广泛应用,成为了事实上的软件过程改进的工业标准。

CMM为软件组织描述了从混乱的、不成熟的软件过程向成熟的、有纪律的软件过程改进的一条途径,共包含了5个成熟度级别(图1)。除了级别1,每个成熟度级别都包含了几个关键过程域,这些关键过程域确定了实现一个成熟度级别所必须解决的问题。

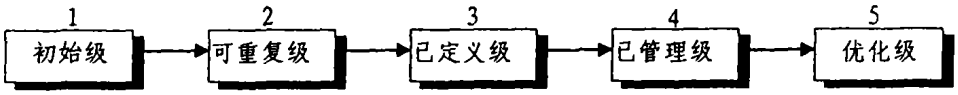

图1 软件过程成熟度的分级标准

CMM关键过程域	符合程度	级别
过程变更管理	0%	5级
技术改革	0%	
缺陷预防	35%	
定量过程管理	10%	4级
软件质量管理	10%	
同级评审	80%	3级
组间协调	20%	
软件产品工程	70%	
集成软件管理	55%	
培训程序	15%	
组织过程定义	25%	
组织过程聚焦	5%	
软件配置管理	80%	2级
软件质量保证	75%	
软件转包合同管理	35%	
软件项目跟踪监督	60%	
软件项目计划	75%	
需求管理	70%	

图2 V-Model基于CMM关键过程域的评价

自CMM被提出来后,围绕着以CMM为基础,评估组织的软件过程能力并进而改进组织的软件过程的研究已经有很多。CMM虽然指出了组织为达到某个能力成熟度级别要做的事情(关键实践),但并没有给出达到这些目标的具体的操作步骤。有鉴如此,不少研究机构提出了一些体现CMM思想的过程模型,来具体指导组织的软件过程改进。

V-Model是德国政府资助研究的过程模型,是德国联邦政府IT系统的开发标准,同时也是奥地利和瑞士相应标准的基础。V-Model关注的是项目级的软件过程,比较好地体现了CMM二级的关键过程域。图2是对V-Model基于CMM关键过程域的评价[1](第二栏的百分数表明V-Model可以达到CMM某个关键过程域的程度)。

本文首先关注的是V-Model如何实现可操作的项目级软件过程的过程改进,同时提出一种建立在项目级软件过程基础上的组织级的软件过程模型,并探讨了要实现具体可操作的组织级软件过程改进要做的一些工作和一个实现原型。

下面介绍V-Model的模型结构与思想,然后介绍我们提出来的以V-Model作为基础的体现CMM思想的组织软件过程模型,最后对本文作出总结。

2 V-Model的体系结构与实现思想[2]

V-Model是在欧洲比较流行的IT系统的开发标准,它规定了系统开发时必须要做什么活动,活动要如何完成,以及为完成某个活动要使用的工具。V-Model包括:

·生命周期过程模型(the Lifecycle Process Model)

·方法的分配(the Allocation of Methods)

*)本文得到国家863项目支持,项目编号:863-306-ZD12-02-2。

·功能工具需求(the Functional Tool Requirements)

2.1 三阶段标准的概念

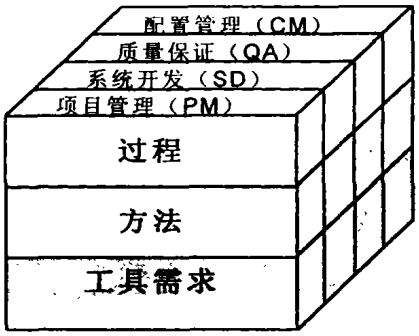

图3 标准的三层结构

此标准包括三级：

1 过程(Procedure)："要做什么事情?"

这一级规定了在系统开发过程中必须要做什么活动，在这个过程中将产生什么结果以及这些结果中必须要包含什么内容。

2 方法(Methods)："事情要怎么做?"

决定在做第一阶段建立的活动时要使用什么方法以及结果将以何种方法表示出来。

3 工具需求(Tool Requirements)：

"为做某事情必须要使用什么工具?"

在这一级，规定了在系统开发中所使用的工具必须要具备的功能特征。

在所有的级别上，规则按照活动域，即所谓的子模型来组织，内容包括：项目管理(project management，PM)；系统开发(system development，SD)；质量保证(quality assurance，QA)；配置管理(configuration management，CM)。

2.2 过程模型

此过程模型由四个子模型组成，这些子模型彼此紧密联系相互影响。

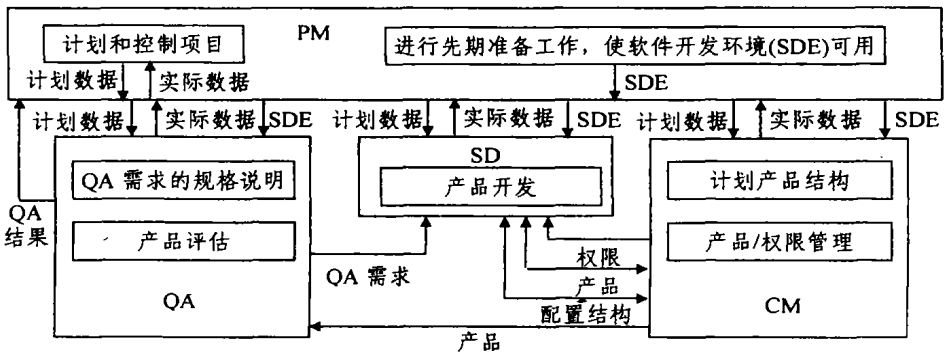

图4 子模型间的协作

PM：计划、控制与协调SD、QA、CM子模型。

SD：开发系统或软件。

QA：详细规定质量需求、测试用例和标准，检查产品和标准的一致性。

CM：管理所产生的产品。

每个子模型由一些按一定顺序流程组织的主要活动以及活动产生的中间(最终)产品/文档组成。

具体使用V-Model时，必须要把任务(活动)分配给各个人，这可以通过角色来实现。

将角色分配给过程模型的活动可以通过一个分配矩阵来实现，而一个人可以被分配几个角色。

在每个子模型中，存在：

·一个经理：建立子模型活动的环境并作为高层决策的决定者。

·一个负责人：计划和监测子模型的活动。

·一个或多个组员：按子模型已计划的任务工作。

在应用过程建模时，所有合理的以及必须的开发文档被作为系统或软件开发的样例而被详细指定，这为实际剪裁标准过程提供方便。

2.3 方法分配与功能工具需求

V-Model的方法分配与功能工具需求使此模型具有实际指导软件过程改进的价值。

2.3.1 方法分配 方法分配以一种普遍可接受的方式规范了方法的应用。为达到这个目的，方法分配的定义仅限于基本的方法。这些基本的方法可以在商业化的工具上实现。基本方法是指描述系统的一个特定方面(如功能、数据定向)或系统开发的某个特定部分(如分析、初步设计)的系统过程(methodical procedure)。基本方法的例子有：E/R建模，状态转换建模。

如何使用方法分配：首先必须为某个项目选择合适的活动与产品/文档，也就是一个剪裁的过程。在项目中执行的每个活动，将会建立相应的方法。方法分配(Methods Allocation)文档里有分配表，实现方法到活动的映射。同时，在方法分配文档里，有每个方法的具体的使用方法。另外，方法分配文档里有一个界面(interface)部分，描述了方法与方法之间的信息交流。

2.3.2 功能工具需求(工具标准) 工具标准的应用有很多优点，比如可以减少工具选择的风险，已知的工具功能可以保证可计算的生产率，等等。当把工具用于一个软件开发环境(SDE)中去时，规范的标准目录可以有助于指定所使用的工具必须具有什么功能标准。

软件开发环境(SDE)是指在一个项目中用于开发IT系统的硬件、软件和组织的管理措施。SDE的参考模型将SDE所提供给用户的所有IT服务放在一个基本模式里，这样一个参考模型的基本组件就被称作服务单元(service units)。一个SDE参考模型的服务单元是指从用户角度组织SDE的软件

部分，代表了工具需求的要点。通过为每个服务单元定义实际的工具需求可以实现这一点。V-Model 里有每个单独的服务单元的详细的功能需求。图5显示了需求、服务单元和工具之间的关系：

为了安装一个真实的 SDE，所使用的工具必须满足服务单元的需求。

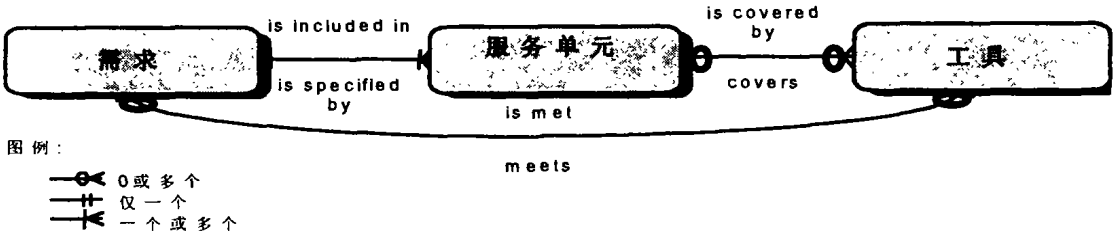

图5　需求、服务单元与工具之间相互关系

相应于过程模型的软件开发环境(SDE)是通过参考模型来描述的。SDE 参考模型通过生命周期过程模型和服务单元来得到精确的定义。

2.4　模型评价

V-Model 通过方法分配和功能工具需求，比较清晰地定义了项目级的软件过程改进的具体的操作流程。

但 V-Model 主要针对的是单个项目的软件过程改进，也就是说主要体现的是 CMM 二级以及与项目级相关的 CMM 三级的几个关键过程域，较少体现与组织级相关的 CMM 关键过程域。我们可以在项目级过程模型基础上，建立组织级的软件过程模型。

3　基于 CMM 和 V-Model 的组织软件过程模型 COPM

由图1可知，V-Model 对于 CMM 二级的关键过程域体现得比较好，而对于 CMM 三级，除一些关键过程域(同级评审，软件产品工程，集成软件管理)外，对于关键过程域组间协调、培训程序、组织过程定义和组织过程聚焦就没有很好地实现。这是可以理解的，因为 V-Model 聚焦的是一个项目范围内的软件过程改进，而 CMM 三级及更高级关注的是组织范围内的过程提高。下面首先介绍 V-Model 中较少体现的 CMM 三、四级的有关关键过程域的内容。

3.1　CMM 三、四级的一些关键过程域

组织过程聚焦的内容主要是了解项目和组织的软件过程状况，协调有关软件过程的各项活动(如软件过程的制定、维护、评估和改进)(图6)。

组织过程定义指由负责软件过程活动的组(如软件工程过程组 SEPG)在组织层上定义软件过程，包括制定和维护组织的标准软件过程以及相关过程描述信息。这些过程描述信息包括：组织标准软件过程(软件过程体系结构和软件过程元素)；对批准使用的软件生命周期的描述；对于剪裁组织标准软件过程的指南和准则；组织的软件过程数据库；与软件过程有关的文档库。

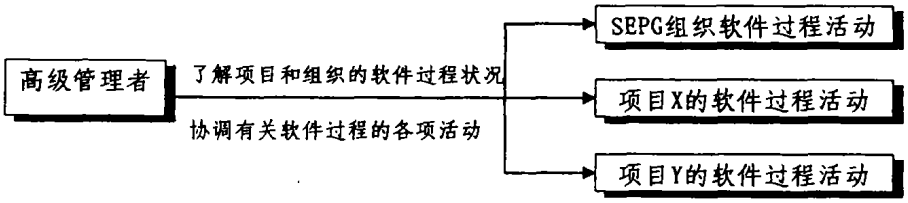

图6　组织过程聚焦

培训程序主要的过程是：首先根据每一个项目当前及其将来对技能的需要，正确判断组织、项目及个人所需的培训，再开发或完成包含这些内容的培训。

组间协调包括软件工程组与其它工程组一起参与阐述系统级的需求、目标和问题，以及计划和管理组间协作的技术界面和交互行为，以保证整个系统的质量和统一性。

对于 CMM 三级的这四个关键过程域，我们可以建立一个子模型：组织管理(Organizational Management，OM，图9)来体现它们的内容。

CMM 四级的两个关键过程域是：定量过程管理和软件质量管理，聚焦于使用定量分析技术(特别是统计的技术)来控制软件过程，来消除过程变化的某些特定的原因[3]。对于这两个过程域，我们建立一个子模型：定量管理(Statistical Management，SM，图10)来体现它们的内容。

我们所做的平台：基于 CMM 的软件质量保障平台的主要目标是在平台中体现 CMM 二、三、四级关键过程域的思想。

3.2　组织软件过程模型

我们提出的组织软件过程模型的三层结构如下图所示：

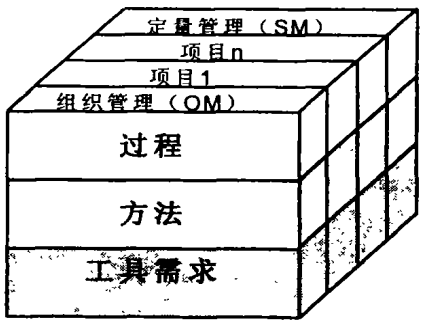

图7　COPM 模型的三层结构

其中：

·过程：做什么

·方法：怎么做

·工具需求：在确定了要做什么与怎么做后，要使用具有什么功能特征的工具来完成活动。

在所有的级别上，规则按照子模型来组织。

具体的组织级的软件过程模型如图8所示：OM 协调与控制组织里的各个软件项目的活动，而 SM 定量管理各个软件项目过程。

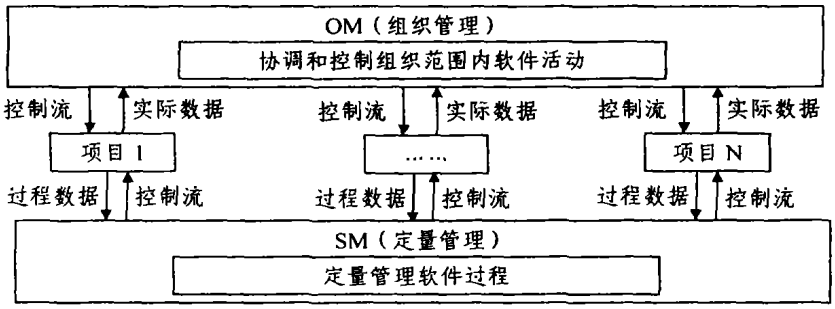

图8　组织软件过程模型(COPM)

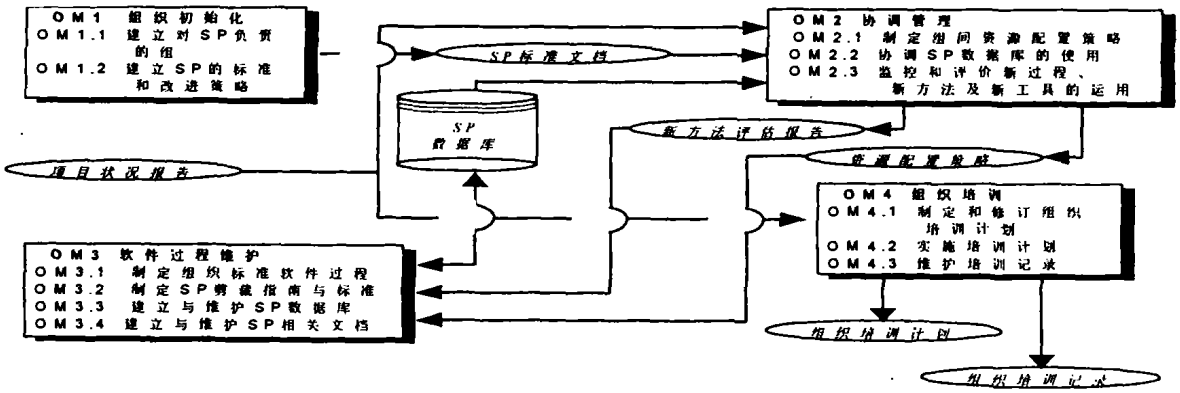

图9　组织管理(OM)子模型

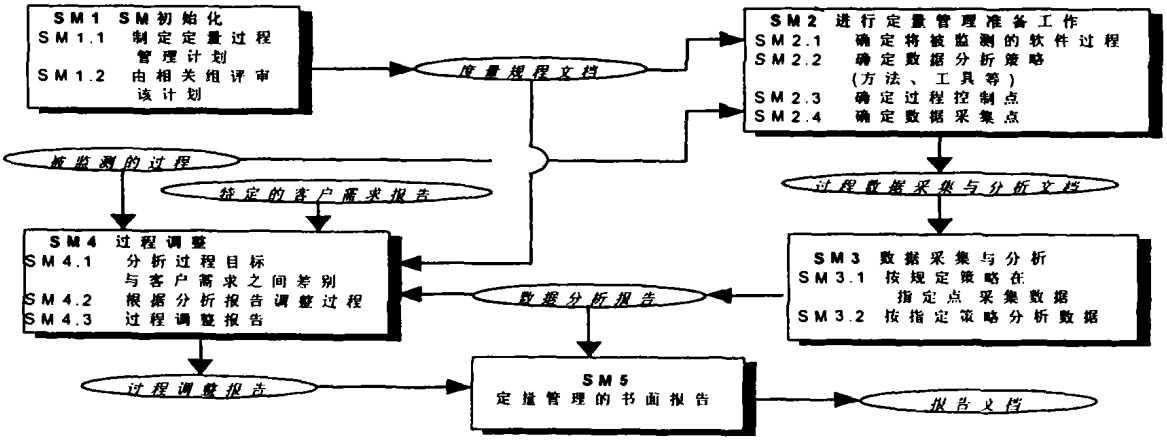

图10　定量管理(SM)子模型

下面的两张图具体刻画了组织管理与定量管理子模型中的子活动与子活动产生的产品/文档。(图9中的 SP 代表软件过程)

组织管理(OM)子模型由四个主要的子活动组成：1)组织初始化：建立对软件过程负责的组及制定软件过程标准和改进策略。2)协调管理：制定组间资源配置策略，协调软件过程数据库的使用。3)软件过程维护：制定组织标准软件过程和软件过程剪裁指南与方针。4)组织培训：根据各个项目状况报告制定与修改组织培训计划，实施培训和维护培训记录。

定量管理(SM)子模型由五个主要子活动组成：1)SM 初始化：主要是制定定量管理的计划。2)进行定量管理准备工作：主要是确定将被监测的软件过程、数据分析策略，以及过程控制点与数据采集点。3)数据采集与分析：采集数据并按一定策略分析。4)过程调整：按所分析的数据和客户的需求报告调整过程并完成过程调整报告。5)定量管理报告：写出定量管理的书面报告。

实际使用此模型时，必须安排特定的人员负责特定的子活动，即实现角色与活动的映射。对于每个子模型，我们可以建立一张角色--活动表，来完成这种对应关系。

对于每个子模型，组织必须要安排一个负责人，由他建立子模型活动所需要的环境并负责计划与监控子模型的活动；同时要安排相应的组员来负责执行相应的子活动。

同时，为了以后进行过程剪裁的方便，模型应该包含有每个子活动所产生的产品/文档信息。这些产品/文档信息表明一个组织按照一个标准的软件过程运行所产生的所有中间以及最终产品/文档资料。

上面所讨论的只是完成了对应于 V-Model 的过程级(Procedure)，即为体现 CMM 的某些关键过程域，必须要做什么事情以及这些事情的先后顺序。要使一个模型可用，必须要指明为完成某个子活动，要采取什么措施(相应于 V-Model 的方法级)以及要使用的工具的所应具有的性质(相应于 V-Model 的工具需求级)。

3.3 方法与工具需求的实现

本节讨论我们提出的这个模型中的方法与工具需求的实现问题。

3.3.1 *方法的实现* 在项目中执行的每个活动，应该建立相应的方法。我们可以设计类似于 V-Model 的方法-活动分配表，实现方法到活动的映射。

方法结构：

每个方法应该具有一些共同的结构，比如：1)标识/定义；2)方法主要特征；3)方法应用的说明(方法如何被应用到活动中去)；4)界面(描述了和别的方法的界面)；5)其它的描述。

子活动 SM 3.2: 按指定策略分析数据
产品: 数据分析报告
参考方法: 方法1; .....; 方法N

图11 方法分配表

3.3.2 *工具需求的实现* 要想使软件过程模型得到实际的应用，首先必须确定过程应用的软件开发环境，也就是各个具体项目级的软件开发环境。在这里，最重要的是建立每个服务单元的详细的功能需求(服务单元是指从用户角度组织 SDE 的软件部分)。

3.4 在平台上实现此模型的探讨

我们可以在基于 CMM 的软件质量保障平台 CPMS 中实现此模型：

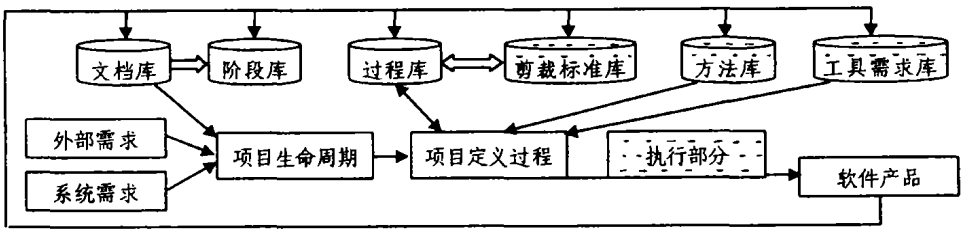

图12 基于 CMM 的软件质量保障平台框架(定义部分)扩充图

对图12稍作说明。本图主要是完成特定项目的软件过程的生成，包括按照需求和剪裁规则来决定特定的项目软件过程所包含的活动，和方法库交互决定完成这些活动所使用的方法，和工具需求库交互决定项目的软件开发环境等。加点部分的三个库是我们对原先平台的扩充，以便平台可以体现我们提出的软件过程。

剪裁标准库中存放用于指导剪裁标准软件过程的一些基本的规则文档，可以指导如何生成特定的软件过程；方法库中存放为完成特定活动所推荐的方法的汇总。方法分配表存放于此库中，以完成活动-方法的映射；工具需求库中则存放有每个服务单元的详细的功能需求。

具体运行时，首先根据需求说明，确定欲采用的软件周期模型，不同的软件生命周期模型会有不同的阶段模式。再根据组织的不同状况选择合适的阶段模块。用户可以随时扩充和裁剪阶段模块库以适应其需求。接着根据这些信息进行特定项目的软件过程的生成。注意这是一个反馈的过程，用户可以对这些库进行创建、修改、删除等操作，其本身也可以从完成的项目中吸取经验教训，进行动态优化。

下面对另外三个库作简要的说明：

1)文档库：主要用于组织和维护各种文档模板和完整的模板样例。

2)阶段库：事先将完成好的标准阶段模块存入库中，用户可以直接根据需要调出这些阶段模块；选择合适的模块后再略加修改便可以生成自己的阶段模块。

3)软件过程库：存放针对不同周期模型的软件过程描述。

基于 CMM 的软件质量保障平台的执行部分的结构图如下：

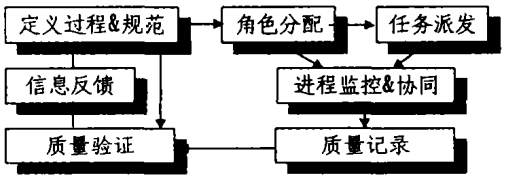

图13 软件质量保障框架执行(管理)部分功能结构图

定义部分完成了特定项目的过程生成，包括子活动、文档、对应于子活动的方法以及软件开发环境的描述。执行部分将完成将各个子活动分配到具体的个人，最终由各个人协作完成特定的项目活动。

保障平台的执行管理部分主要有监督和验证两方面的职能。监督可以帮助组织按照制定的过程和质量标准开发软件，对软件开发的进度和中间产品有一个良好的维护和控制，保证最终软件产品的。验证帮助组织了解质量记录的完整性以及同标准的一致程度，同时可以产生必要的反馈，验证所订标准的合理性和完善程度。

结论与评价 本文探讨了一种基于 CMM 思想的组织级的软件过程模型，并提出了所要解决的一些问题，并讨论了此过程模型在软件质量保障平台中的实现。但必须指出的是：软件过程是一个与人有很大关系的东西，很多的过程活动是不能够仅由平台来体现的。比如组织管理子模型中的协调管理子系统，实际上很大程度上依赖于人的因素；又比如定量管理子模型中的一些活动。实际应用时必须考虑这一点。

自 SEI 提出 CMM 后，已经有很多组织与企业应用

(下转第138页)

象伪程序，其实它的语义恰如状态的替换，Abrial 已经证明了这种赋值和 VDM 中前置/后置条件功能是相等的，如：

```
VDM:
      sort ::
        wr list:seq(TYPE)
      pre true
      post is_sorted(list)&
           is_permutation(list,list')
B:
    sort ::
    ANY newlist WHERE
          is_sorted(newlist)&
          newlist :perm(rng(list))
    THEN
           list := newlist
    END
```

(4)*前置条件、不变式和证明法则*　Z、VDM 和 B 在前置条件和不变式的使用上都有不同的方法，在 Z 中，前置条件没有明确的陈述，它可从 delta 模式的定义中计算得到。

在 VDM 和 B 中，前置条件是明确陈述的，冗余的前置条件需要一致性检测，即这些前置条件是这个操作所必需的吗？

B 和 VDM 的不同之处在于它们处理不变式的方法上，在 VDM 中不变式被假定为任何前置或后置条件的一种暗示，因此关于该操作唯一要证明的就是其可行性。

在 B 中，不变式是冗余的，不变式的存在与否都不会改变操作的定义，它并不是操作后置条件的一部分，因而需要证明该操作是否必须保留不变式，而在 VDM 中则应该明确无误地保留。

(5)*语言应用范围*　Z 是一种相当严格的规格说明语言。

VDM 和 B 是"广谱"的，在两类语言中，包含了必要的程序构造。

B 包含了一个很小的语言子集，分6个构造部分，所有的数据类型被封装在抽象机器中，这就保证了精化到代码的过程都在一个统一的语义框架上进行。

(6)*结构化规格说明和实现*　B 比 VDM 有着更强大的结构化构造，而且也很不同于 Z 的模式演算，B 是以对象为基础，包含有各个种类的信息被增强了，因此操作的状态封装很易实现，然而还缺少真正意义上的继承和多态性等相关概念，所以它还不是太"面向对象"的。

B 方法的分层开发的观念是非常强有力的，从而允许一个复杂的开发被分解为一系列小的构造，以此满足工业化应用系统的精化需求。

(7)*工具支持*　B 因为有工具支持而获得相当大的优势，象 B－Toolkit 这种高质量的商业软件有一套完整的开发过程的支持用来完成如下一些任务：类型检测、动画、证明法则生成、交互或自动的证明支持、代码翻译、文档以及带有配置管理功能的综合开发环境支持。

结束语　上文中，我们简要介绍 VDM、Z、B 三种语言和方法及其优缺点，并从七个主要方面作了分析比较，作为总结，表1从基础、开发阶段等五个方面对上述三种规格说明方法进行比较。

表1　B 与 VDM、Z 的比较

属性	VDM	Z	B
基础	部分函数、集合论	谓词演算、集合论、模式	最弱前置条件、集合论
开发阶段	规格说明、设计	规格说明	规格说明、设计、实现
风范	前置/后置条件、函数	模式符号表示、关系	严格的程序设计语言
工具支持	在规格说明级	在规格说明级	所有开发阶段
培训支持	图书、课程	图书、课程	实例研究、课程

从上表可以看出，B 相对于 Z 和 VDM 来说，有很大的优越性，再加上商业工具的大力支持，因而在英、美、法等国得到了较广泛的应用，然而 B 语言还存在一些不足之处，下一步我们还将就 B 语言的扩充做进一步深入的研究，希望 B 语言和方法在国内也能得到广泛、深入的应用。

参 考 文 献

1　袁晓东，郑国梁. Z 的面向对象扩充 COOZ 的设计. 软件学报，1997(9)：694～700

2　袁晓东，许皓，胡德强，李勇，郑国梁. 现有 Z 面向对象扩充语言的比较. 计算机科学，1997(3)：58～61

3　Lano K. The B Language and Method. Springer-Verlag London Limited. 1996

(上接第127页)

CMM 来评估并且改进它们的软件生产过程，也有很多研究 CMM 实践的文章发表，总结 CMM 在实践中的经验与教训[5,6]。文[5]的研究表明，有效的过程评估(以及相应的提高)的模型/方法必须要追求组织所有方面的一致性的改进而不仅仅是工程方面的改进。要实现组织范围内的软件过程改进，必须要综合考虑其中的各种因素。

必须强调的一点是：CMM 描述的是过程应当解决的问题而非应当如何去执行，因此在过程改进中需要确定很多"执行细节"[7]。由于 CMM 重点在于软件问题，其他一些属于整入进大纲范畴的问题，在 CMM 中仅仅是提以而已。这些问题包括战略业务规划、建立产品线、采用有效技术以及人力资源管理等。

参 考 文 献

1　Schuppan V，Rußwurm W. A CMM-Based Evaluation of the V-Model 97，Software Process Technology，7th European Workshop，EWSPT2000，Springer

2　V-Model--Brief Description. http://WWW. v-modell. iabg. de

3　Paulk M C. Toward Quantitative Process Management With Exploratory Data Analysis，1999 International Conference on Solftware Quality http://WWW. sei. cmu. edu

4　杨一平，等. 软件能力成熟度模型 CMM 方法及其应用. 人民邮电出版社，2001

5　Cattaneo F，Fuggetta A，Sciuto D. Pursuing Coherence in Software Process Assessment and Improvement，Software Process Improvement Practice 2001，John Wiley & Sons，Ltd.

6　Blanco M，et al. SPI Patterns：Learning from Experience，European Software Institute，IEEE Software，May/June 2001

7　SEI：The Capability Maturity Model：Guidlines for Improving the Software Process