ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.8, August 2006, pp.1669−1680
DOI: 10.1360/jos171669

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种基于 PDCA 的软件过程控制与改进模型*

武占春[1,2+], 王 青[1], 李明树[1,3]

[1](中国科学院 软件研究所 互联网软件技术实验室,北京 100080)

[2](中国科学院 研究生院,北京 100049)

[3](中国科学院 软件研究所 计算机科学重点实验室,北京 100080)

A PDCA-Based Software Process Control and Improvement Model

WU Zhan-Chun[1,2,+] WANG Qing[1], LI Ming-Shu[1,3]

[1](Laboratory for Internet Software Technologies, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

[2](Graduate School, The Chinese Academy of Sciences, Beijing 100049, China)

[3](Laboratory for Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

+ Corresponding author: Phn: +86-10-87392220, E-mail: wzchun@itechs.iscas.ac.cn

Wu ZC, Wang Q, Li MS. A PDCA-based software process control and improvement model. *Journal of Software*, 2006,17(8):1669−1680. http://www.jos.org.cn/1000-9825/17/1669.htm

Abstract: CMM/CMMI (capability maturity model/CMM integration) has been accepted since 1999 by Chinese software organizations, and widely used since then. But there are limited number of CMM/CMMI users in China. By launching a survey, problems of using CMM/CMMI are identified, and the negative impact is analyzed. Based on the analysis, a PDCA(plan-do-check-action)-based software process control and improvement model is presented, and the SoftPM which is based on the model has been developed. The SoftPM has been widely used in China, which proves the effectiveness of the model to solve the problems identified in the survey. Software process effectiveness and efficiency are also improved for those software organizations that are using or will use CMM/CMMI.

Key words: software quality assurance; project management; software measurement and analysis; software quality management; software process improvement; CMM (capability maturity model); CMMI (CMM integration)

摘 要: CMM/CMMI(capability maturity model/CMM integration)自 1999 年开始为中国软件企业所接受并逐步得以推广,但目前中国实施 CMM/CMMI 的企业还不多,有些企业实施效果并不理想.通过调查软件企业在实施 CMM/CMMI 过程中存在的问题,并对发现的问题及其负面影响进行分析,提出了基于 PDCA(plan-do-check-action)的软件过程控制与改进模型,开发了 SoftPM 软件质量管理平台.该平台的广泛应用表明,该模型对提高 CMM/CMMI 的企业软件过程的效率和改善实施效果很有帮助.

关键词: 软件质量保证;项目管理;软件度量分析;软件质量管理;软件过程改进;CMM (capability maturity

* Supported by the National Natural Science Foundation of China under Grant Nos.60273026, 60473060 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2004AA112080 (国家高技术研究发展计划(863))

Received 2005-10-20; Accepted 2005-12-31

model);CMMI (CMM integration)
中图法分类号: TP311 文献标识码: A

20世纪80年代末,借鉴制造业通过控制和改进工艺流程提高产品质量的方法,软件工程界提出了通过控制和改进软件过程来提高产品质量的思想[1].此后,以过程管理为基础的规范化管理方法不断涌现,如 SPICE (software process improvement and capability dEtermination),CMM(capability maturity model),ISO9000:2000版等[2].其中美国 SEI(software engineering institute)软件工程研究所于 1991 年提出的能力成熟度模型 CMM 在全球软件业得以广泛应用并收到很好的效果[3,4].2004 年,SEI 推出了集成多种 CMM 模型(如软件 CMM、系统工程CMM)的CMMI(CMM integration).这个模型采用连续和分级两种表示方式,为实施过程改进提供了更大的灵活性[5],因此受到越来越多的软件组织的欢迎.

我国政府已意识到,通过实施 CMM/CMMI 促进软件企业规模化、规范化、国际化,是我国软件产业发展的重要途径,并出台了相关鼓励政策.与此同时,我国软件企业也意识到规范化管理的重要性.因此,近年来有越来越多的软件企业采用 CMM/CMMI 建立企业软件过程、规范项目开发.尽管如此,我们的调查显示,由于一些软件企业软件过程的实施效果不佳,其负面效应制约了 CMM/CMMI 在我国软件企业的大规模应用.

建立一套符合 CMM/CMMI 要求的软件过程不难,难在如何有效实施定义好的软件过程上.这是因为:第一,软件组织和软件开发项目处于一个动态的环境中——技术不断发展、市场不断变化、人员经常流动,这要求软件过程随之动态变化,且又要处于受控状态;第二,任何一种规范管理体系的实施(如基于 CMM 的软件过程)都意味着工作方式的变革,员工能否自然地接受这样的变革决定了软件过程能否真正得以实施;第三,规范化管理需要增加管理成本投入以满足模型的要求,如配备质量保证人员监控软件过程的运行.因此,有效地实施软件过程应具备以下几个条件:(1) 软件过程能够及时调整、改进,以适应不断变化的外部环境;(2) 实现人与过程的自然融合,以使流程得到有效遵守;(3) 能够对软件过程进行度量,以便为控制和改进软件过程提供客观依据;(4) 有自动化系统的支持,以提高软件过程的执行效率,降低管理成本.

近些年来,围绕软件过程技术、项目管理、软件度量的研究以及软件企业实践经验总结,都为指导软件组织提高软件过程实施效果做出了有益的贡献,但在解决动态环境下对软件过程的控制和改进方面都还存在着一定的局限性,人——软件过程主体之一的参与问题也未得到有效解决.

本文提出了基于 PDCA(plan-do-check-action)的软件过程控制与改进模型,并开发了基于该模型的 SoftPM 平台.到目前为止,该平台已在 100 多家软件企业以及我国主要软件园区和软件专业孵化器使用,有效地帮助软件企业在实施 CMM/CMMI 过程中提高管理效率、降低管理成本,增收节支效果显著.

1 相关研究

为了解决软件过程有效实施的问题,软件工程研究和实践者们在过程建模、项目管理、软件度量、过程改进等方面进行了卓有成效的探索.

Leon Osterweil 首先提出“软件过程即软件”的思想[6].此后,软件过程建模方法和以过程为中心的软件工程环境(process-centered software engineering environment,简称 PSEE)的研究有了较大发展.软件过程建模的目的有两个:其一是采用过程建模语言描述软件过程.描述的方式一般有工作产品、角色和活动 3 种主要方式,其中后一种方式在软件组织实际定义软件过程时使用较为普遍[7,8];其二是软件过程模型实施.软件过程模型的实施通常是在 PSEE 系统支持下实现的,即认为软件过程模型是一段程序,描述为一组预定义的活动,PSEE 系统则可以被看作是运行该程序的环境或者虚拟机,而人作为实施活动的主体被“约束”到软件过程中.但软件组织和软件项目处于一个动态的环境中,软件过程通常难以按照预定义模式来执行.因此,尽管现有的方法对软件过程模型的建立与实施提供了有力的支持,但是这些方法很少在实际的软件项目中得到应用[9].

项目计划是开展项目各项工作的基础,依据计划对项目工作跟踪,可以及时发现并纠正偏差,确保项目顺利完成.同时,为了适应项目外部环境的变化,项目计划的内容也可作相应的调整.美国项目管理学院(Project

Management Institute,简称 PMI)提出的项目管理知识体系(project management body of knowledge,简称 PMBOK)[10]为项目管理提供了较为系统的方法.体系中的项目计划方法通常是面向产品的[11],计划的任务要与中间或最终工作产品相关联.但这样做的问题是,执行任务的负责人只关注能够开发出产品的工程过程,容易造成管理与技术的脱节.避免出现这样问题的方法是,由过程改进的专门机构,如 SEPG(software engineering process group,即软件工程过程组)对全体成员培训,直至项目相关成员完全理解软件过程,并能够在项目中灵活使用[12].但这样一来,软件过程实施效果又过分取决于人员能力、对过程的理解和支持程度.

软件过程度量是控制和改进软件过程的重要手段.GQM(goal-question-metrics)[13]方法是目前使用比较广泛的软件过程度量方法之一.该方法依据组织的商业目标建立度量指标体系,然后在项目执行过程中收集和分析数据.但通过这种方法建立的指标体系是不收敛和不重复的,在实际指导软件企业度量时存在一定的困难[14].

CMM 本身即源自美国政府和军工软件企业的一些成功实践,因此这些软件企业在有效实施软件过程方面有着丰富的经验.美国 Motorola、休斯、波音等公司的 SEPG 立足自身多年的经验积累,为有效实施软件过程、提高产品和项目控制能力提出了很多好的建议[15−17].比如,建立承诺机制、确保员工积极参与到软件过程中来等.日本和印度也是当今国际软件业两个非常重要的国家.自美国人 Edward Deming 把全面质量管理理念带入日本后,在日本掀起了质量热潮,质量控制小组遍及全国.在软件过程改进方面,日本软件业采用了类似的组织——JASPIC,旨在跨越公司界限,分享过程改进的经验[18].印度在实施 CMM 方面取得了极大的成功,其软件业借助规范的管理进入国际市场,并逐步发展壮大,CMM 作者之一 Bill Curtis 将印度软件业过程改进的成功归于“一批纪律性很强的聪明的工程师,和对过程有着深入理解的管理者”[19].我国软件业发展历史不长,对软件过程的理解和认识程度还不高,以上经验虽然值得借鉴,但与我国软件企业实际情况差异较大,指导性不强.

截至 2004 年底,我国共有 156 家软件企业进行了 211 次 CMM/CMMI 评估.相比我国众多的软件企业而言,实施 CMM/CMMI 的软件企业不多,约占全国软件企业的 2%[20].

为了探索制约 CMM/CMMI 实施的因素,我们针对实施 CMM/CMMI 的软件企业进行了调查[20].调查中发现,虽然企业通过实施CMM/CMMI取得了一些收益,如促进了规范化管理、提高了项目控制能力和产品质量等,但存在的问题也较为突出.流程繁琐僵化、实施成本高是两个主要问题.

调查还发现,造成流程繁琐的原因是很多企业的软件过程只关注 CMM/CMMI 模型要求,而不注意符合企业实际情况.这样的软件过程尽管可以通过高层管理者强制推广开来,但由于缺乏有效的手段,如灵活的裁剪和度量分析、问题反馈分析机制等,不能及时调整和改进过程,使得过程繁琐且僵化,无法适应软件企业所面对的动态环境.执行这样的软件过程,要么付出很高的代价,要么开发人员绕开管理过程,使得管理与技术脱节.无论是哪种情况,实施效果都不尽如人意.因 CMMI 实施效果不佳而形成的过程改进“不成功”经验在软件产业内产生了负面效应,降低了其他软件企业实施 CMM/CMMI 的积极性.

调查数据显示,中小型软件企业需要投入年收入的 5%~13%.随着各地政府已经降低甚至暂停了对实施 CMM/CMMI 的奖励,中小型软件企业无力承担实施 CMM/CMMI 的各项费用.而另一方面,我国大型软件企业数量不多,且不少已经通过了 CMM/CMMI 评估.而没有中小型软件企业广泛参与 CMM/CMMI 的实施,扩大 CMM/CMMI 在我国软件业的实施规模、促进我国软件企业全面走向规范化发展轨道是不可想象的.

综上所述,虽然在软件过程建模方法和 PSEE、项目管理、软件度量分析等方面的研究和实践取得了很大进展,但对于解决我国软件企业在动态环境中有效实施软件过程的问题依然存在不足.

2 基于 PDCA 的软件过程控制与改进模型

早在 20 世纪 30 年代,美国的 Shewhart 提出了 PDCA 循环的概念,将提高产品质量划分为 4 个阶段,即“策划(plan)-实施(do)-检查(check)-处理(action)”.4 个阶段不断循环则质量不断提高,而且在 PDCA 的各个阶段内部还能进行 PDCA 小循环,解决该阶段的问题.后来,美国的 Edward Deming 对 PDCA 理论进行完善,包括补充了执行 PDCA 的步骤、帮助发现与分析问题产生原因的 7 种工具(检查单、趋势图、直方图、控制图、因果分析图、散列图和 Pareto 图)等.完善后的 PDCA 理论在企业的质量管理中得到了广泛的应用.同时,PDCA 也成为使任何

一项活动有效进行的一种合乎逻辑的工作程序[14].

本文遵循 PDCA 的原则,提出了基于 PDCA 的软件过程控制与改进模型.模型采用面向过程的方式制订项目计划,利用度量数据的分析对软件过程的执行进行控制,并对组织标准过程进行改进.如图 1 所示.

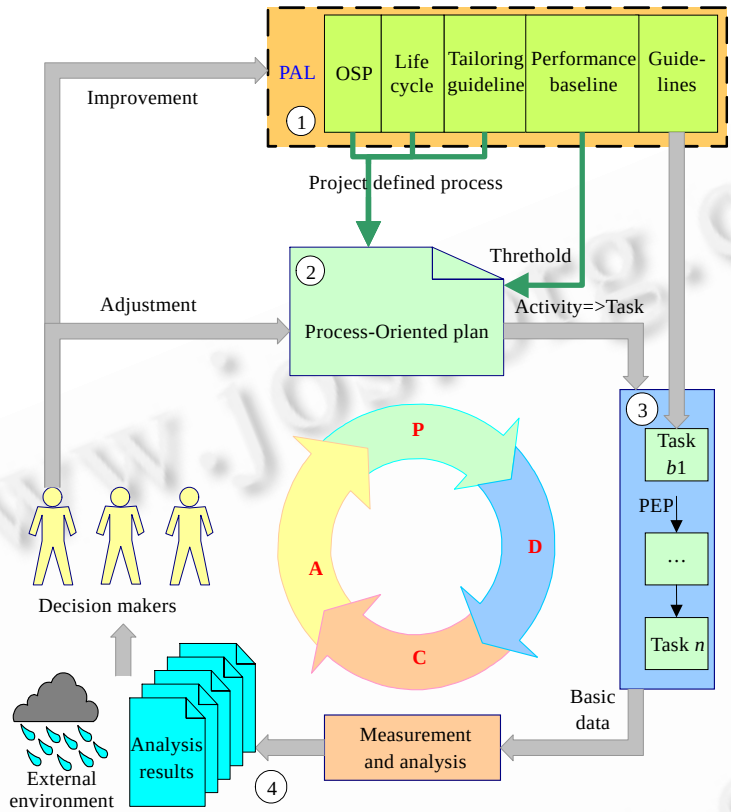

Fig.1 PDCA-Based software process control and improvement model

图 1 基于 PDCA 的软件过程控制与改进模型

P 阶段(plan):项目计划是开展项目工作的基础.为此,模型通过面向过程的计划方法将组织软件过程中的各项活动转化为计划中需要完成的一系列任务,并设定项目控制阈值(包括项目计划控制基准和软件过程性能基线),以便在阶段 C 判断项目过程的执行是否正常.项目计划中的任务落实到执行者,所以这种计划的方法能够确保软件过程活动得以执行.而且,通过调整项目计划中的任务,又提高了软件过程执行时的灵活性,见图 1 中的①和②.

D 阶段(do):任务执行者按照计划、任务指南和规范要求来执行任务,并通过任务报告提交相应的度量数据.至此,组织软件标准过程转化在项目中执行的软件过程,并把相关人员“融入”其中.而且,由于规范和指南也与任务直接关联,帮助执行者完成任务,因此能够进一步提高软件过程的执行效果,见图 1 中的③和④.

C 阶段(check):利用度量分析的算法分析软件执行者提交的基础数据,并与项目计划中预先设定的项目控制阈值进行比较.

A 阶段(action):通过与项目基准的比较可以判定项目计划的符合性状况,即是否能按项目计划确定的成本(在软件组织中主要是投入的工作量)和进度要求执行;通过与过程性能基线的比较,可以判定软件过程是否处于稳定状态.如果通过对数据的分析,发现过程的实际执行不能满足计划的要求或过程不稳定,则必须采取措施对项目执行过程进行控制.这些措施一般作为任务列入项目计划中,也就是说,通过对项目计划的调整,实现对项目执行过程的控制,见图 1 中的③.另一方面,通过对度量数据进行分析还能够反映软件过程是否符合软件项

目的实际需要,以及软件过程能力是否满足客户和软件组织自身发展的要求,从而为改进组织的软件过程提供客观依据,帮助实现软件过程改进,见图 1 中的①.

2.1 组织标准软件过程建模

为实施过程管理,软件组织通常都会定义组织级软件过程和指导过程使用指南、规范及数据等,通常称为组织过程资产 PAL,见图 1 中的①.主要包括以下内容:

- 组织标准软件过程(organization's standard software process,简称 OSP).它包含了对软件过程元素的描述,是软件组织内软件项目所要遵循的公共软件过程,是定义项目软件过程和实施度量的基础.
- 裁剪指南.包括一组允许进行裁剪的条件和指导裁剪的准则,用于指导软件项目选择软件生命周期模型,以及根据项目的需要选择组织标准软件过程,即修改或删除组织标准软件过程活动.
- 软件过程性能基线.由软件过程特征值与上限偏差控制限构成.如果软件过程执行时的实际度量值落在上下控制限之内,则软件过程是稳定的.而上下控制限的宽窄和中心线值的大小则是软件过程能力的体现.
- 指南和规范.用于指导具体的活动执行的步骤和方法.
- 历史数据.用于帮助软件项目在依据软件过程制订计划时进行估算以及确定软件过程性能基线.

根据以上描述,可以进行以下形式化定义:

定义 **1**(取元素值操作). 对任意元组 $X=(C_1,C_2,\ldots,C_n)$,若取得 C_i 值,记作 $X.C_i$.

定义 **2**(组织标准过程 **OSP**). $OSP=\{PA_1,PA_2,PA_i,\ldots,PA_n\}$,$PA_i$ 为任意软件过程,是一组活动的集合:$PA_i=\{AC_{i1},AC_{i2},\ldots,AC_{in},PPB_i\}$.$AC_{ij}$ 为构成 PA_i 的一系列活动,且 $AC_{ij}=(type,r,Emetrics,Hmetrics,tg,GL,I,O)$.

- *type* 是活动的类型,与该活动所属的 *PA* 属于相同类型,为工程、过程管理、项目管理或支持活动中的一种.
- *r* 为执行该活动的角色.
- *Hmetrics* 是活动的历史数据,指导制订项目计划时确定进度和工作量.对不同组织有不同的估算模型,如 COCOMO 模型,利用历史数据进行估算,记作 $Es(Hmetrics,AC)$,估算结果为该活动的工作量和进度的计划值$(Effort,Schedule)$.
- PPB_i 为过程 PA_i 的性能基线.$PPB_i\in PPB$,$PPB_i=V\pm m\sigma.V$ 为该软件过程的特征值,例如测试过程的缺陷密度.特征值通过对软件过程中活动的历史数据分析而得到,$V=f(AC_{ij}.Hmetrics)$,f 为对该过程活动的度量算法函数,通常采用的是统计过程控制(statistic process control,简称 SPC)算法.σ为标准差,m 为任意整数.

需要说明的是,对过程实际执行的软件过程的特征值进行度量,若处于该范围内,则过程是稳定的;否则,表示软件过程不稳定,需要采取纠正措施进行控制.项目一般需要根据项目特征有选择地利用 SPC 技术对项目过程进行控制.

- *Emetrics* 是在执行该活动时需要收集的度量指标集合;
- *tg* 为裁剪标志,$tg=1$ 表示可以裁剪,$tg=0$ 表示不能裁剪;
- *GL* 为指导该活动的指南/规范;
- *I,O* 分别表示该活动的输入和输出.

定义 **3**(裁剪操作算子 **TG**). 对$\forall AC$,若该活动的裁剪标志 $AC.tg=1$,执行裁剪操作记作 $TG(C,AC)$,表示对活动 *AC* 进行的裁剪操作.$C=\{C_1,C_2,\ldots,C_n\}$,表示允许裁剪的一组条件,与项目特征相关,如项目目标、采用的技术、项目团队的人员规模等.设项目特征为 T,$T=\{t_1,t_2,\ldots,t_n\}$,则 $T\cap C$ 是项目进行裁剪的依据.

2.2 面向过程的项目计划

传统的项目计划方法是面向产品的[10−12],典型的项目计划方式是先把软件系统分成若干子系统,然后把子系统分解为若干模块,再把模块开发作为任务列入项目计划并分配给相关人员[1].如第 1 节所述,以这样的方式制订项目计划要求项目人员对软件过程的理解较为透彻才能确保软件产品是依据软件过程的要求开发的.但

项目成员对软件过程的理解总是有差异的,因此,面向产品的项目计划方法不能保证软件过程得以有效执行,进而影响软件过程的实施效果.

考虑到无论是软件的最终产品还是中间产品,总是软件过程中某些活动的输出,如果软件过程中的活动都转化为项目计划中的任务,不仅能完成软件产品的开发,还能保证开发是按软件过程的要求进行的.同时,利用对项目计划的跟踪机制,可同时实现对计划符合性及过程性能的监督与控制.因此,本文采用了面向软件过程制订项目计划的方法.也就是先确定在项目中执行哪些过程,并据此进一步确定需要完成的一系列任务(见图 1 中的①,②和③).面向过程的计划分为以下几个步骤:

1) 定义项目过程.即根据项目特征,选择适合项目需要的软件过程及活动.
2) 估算进度和工作量.对过程中的活动进行估算,确定进度和工作量.
3) 分配任务.依据对活动估算的经过确定任务的进度和工作量,并指定负责人.
4) 设定项目计划的控制阈值和软件过程的控制阈值,以便比较计划和过程的执行是否有偏差.

定义 **4**(项目定义过程(**project execution process,**简称 **PDP**)). PDP 是经过对 OSP 裁剪后得到的软件过程活动的集合.PDP 中的活动有两类:一类是保留下来的 OSP 中不允许裁剪的活动,仍记作 *AC*;另一类是 OSP 中经过了裁剪的活动,记作 *AC′*,即 $PDP=\{\ldots,AC_{ij},\ldots,AC'_{mn}\}$.PDP 中的活动是项目执行过程(project execution process,简称 PEP)分配任务的依据.

定义 **5**(项目执行过程 **PEP**). 虽然裁剪指南可以为项目选择软件过程提供一定的灵活性,但裁剪 OSP 后形成的 PDP 可能依然存在不适应项目需要的情况,比如粒度过大或项目情况特殊,无法执行某些不允许裁剪的活动等.因此,项目执行过程需要对 PDP 中的活动细化成任务 *Task* 才能执行,进一步提高了软件过程的适应性.对 PDP 活动细化形成的一系列任务 *Task* 构成了项目执行过程,同时 PEP 中还包括计划控制基准和过程性能基线,用于在执行软件过程时判断执行状况.

$PEP=\{Task_{111},\ldots,Task_{ijk},\ldots,Prj_{th},PPB'\}$. *PDP* 中 AC_{ij} 转化一组任务$\{Task_{ijk}\}$, *k* 为 0 或任意正整数.

$Task_{ijk}=(type,Parent,R,Emetrics,PAttribute,ACAttribute,GL,I',O')$.

- *R* 为任务的执行者.
- *Parent* 是 *Task* 所属的活动的下标号.
- $PPB'\subseteq PPB$,项目根据项目特征选择的一组软件过程控制基线.
- $I'\in I,O'\in O$.
- *PAttribute* 为任务的计划属性,是任务的计划工作量和进度.*PAtribute*=(*effort*,*schedule*),*schedule*=(*Starttime*,*Endtime*),*Starttime* 和 *Endtime* 分别是任务的起止时间.
- *ACAttribute* 为任务执行时收集的计划属性的实际值.
- Prj_{th} 是项目计划控制基准.它的作用是比较项目实际进度和工作量投入是否符合计划要求.设有函数 θ,为计划值与实际值之间的比较算法,如用来比较实际成本和进度与计划成本和进度偏差的挣值分析算法.$Prj_{th}=\theta$(*PAtteibute*,*ACAttribute*).

基于上述定义,确定项目计划形式化表述如下:

算法 **1**. 制订项目计划.

(1) 初始化:$PDP=PEP=P_{tg}=\varnothing$.$P_{tg}$ 为过程记录,用来记录 OSP 中活动在项目中的使用情况.

(2) 对 OSP 进行裁剪.根据定义 2 和定义 3,设项目的特征为 *T*,若 $T\cap C!=\varnothing$,遍历 OSP 中 *PA* 的活动 *AC*:
若 *AC.tg*:=1,则根据裁剪指南对其进行裁剪:$AC'=TG(C,AC)$;$PDP=PDP+AC'$;$P_{tg}=P_{tg}+AC'$;
若 *AC.tg*:=0,则 $PDP=PDP+AC$.

(3) 得到项目定义过程 PDP,$PDP=\{\ldots,AC_{ij},\ldots,AC'_{mn},\ldots\}$.同时,在过程记录 P_{tg} 中记录下软件过程中哪些活动经过了裁剪.即 $P_{tg}=\{\ldots,AC'_{mn},\ldots\}$.

(4) 利用历史数据对项目定义过程中的活动估算工作量和进度.对$\forall AC\in PDP\wedge\forall AC'\in PDP$,根据定义 2,即 *Es*(*AC.Hmetrics*,*AC*)=(*Schedule*,*Effort*),为执行这个活动的进度和工作量.

(5) 确定任务.对$\forall AC_i$和 AC_i',$\exists\langle Task_1,Task_2,\ldots,Task_n\rangle$,取完成任务的一组成员 $r\subseteq T$,令

$Task_j.Parent=i$; //任务所属的活动

$Task_i.R\in r$; //指定任务负责人

$Task_i.PAttibute=(schedule_i,effort_i)$; /*确定任务的计划属性*/且必须满足:

$(\Sigma effort_i=Effort)\wedge(\max(schedule_i)\leq Schedule)\wedge((Task_n.Endtime-Task_1.Starttime)=Schedule)=1$.

(6) 若$\exists AC$,对于任意 $i,Task_i=NULL$,即此活动并未在项目中执行.将此类活动记作 AC_{ij}'' 加入 P_{tg} 中,得到 $P_{tg}=\{\ldots,AC_{ij}'',\ldots,AC_{mn}'',\ldots\}$,即经过裁剪以及虽然不允许裁剪但未使用的活动集合.

(7) 根据定义 2、定义 5,设定项目控制阈值 $Prj_{th},PPB'\subseteq PPB$.

(8) 得到项目执行过程 $PEP=\{Task_{111},\ldots,Task_{ijk},Prj_{th},PPB'\}$.

至此完成了项目计划的制订,也使得组织标准过程中的活动转化为项目可执行的过程,并确定了计划的控制基准以及受控过程的软件过程性能基线.由于软件过程的任务中关联了执行任务的指南和规范 *GL*,且软件过程的活动都通过任务指定了执行者,因此能够保证软件过程得到有效执行.

另外需要特别说明的是,拆分后的一系列 *Task* 在执行过程中不是一成不变的,可以对任意类型的任务进行增、删、改操作,这样可以提高计划的灵活性.

2.3 过程的控制与改进

2.3.1 对项目执行过程的控制

由于本文采用的是面向过程的计划方法,因此,项目管理中常用的对计划的跟踪与监督方式可以用来控制 PEP 的执行,以及在任务的执行者执行任务时,定期提交任务执行情况的报告,如项目周报或日报,并在项目周报包括两方面内容:计划执行情况和对任务的度量基础数据($ACAttribute,Emetrics$).通过对这些数据进行分析,并与项目控制阈值进行比较,以判断软件过程的执行是否符合要求,见图 1 中的④.根据定义 2 和定义 5 可知:若 $\theta(Task_{ijk}.ACAttribute,Task_{ijk}.PAttribute))<Prj_{th}$ 且$|f(Task_{ijk}.Emetrics)-f(AC_{ij}.Hmetrics)|\leq m\sigma$,则项目执行过程的计划符合性良好,软件执行过程也处于稳定状态;否则就需要采取纠正措施.

确定措施纠正项目计划或过程出现的偏差,通常采用如下几个步骤:

a) 利用质量管理的 7 种工具中的 1 种或多种工具的组合,对度量数据进行分析,确定引起偏差的原因;

b) 确定措施或方案,并指定实施者;

c) 实施纠正措施;

d) 跟踪并验证实施效果,如果未能纠正偏差,转到步骤 a).

7 种质量管理工具能够依据数量统计理论,以直观的方式呈现对偏差进行分析的结果,便于步骤 b)中确定纠正/改进措施.图 2 是某项目对缺陷类型进行分析得到的缺陷分布直方图.由图中可以看出,与易用性相关的缺陷(图中 Cosmetic 类型)所占比重过高.这使得开发人员采取加强易用性设计并与客户提早沟通等措施,降低易用性缺陷.类似地,利用质量管理工具也可以帮助分析问题存在的原因,确定过程改进措施.

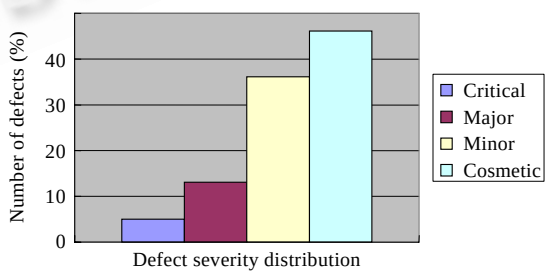

Fig.2 Defect distribution

图 2 缺陷分布

从步骤 a)~步骤 d)的描述还不难看出,采取纠正措施的各个步骤形成了一个 P-D-C-A 循环,步骤 a)到步骤 b)就是一个计划的过程.因此,本文把调整项目计划作为纠正偏差、实现对项目过程控制的主要机制,即通过对图 1 中的②面向过程的计划中任务的调整,实现对软件执行过程的控制.

调整任务包括在项目计划中增加任务,也可以删除、修改.例如,项目进行需求评审时的性能基线是每百条需求缺陷数为 9.5±7.5,而实际度量值为 20.3,表明该过程异常.通过原因分析发现,定义需求的人员与客户沟通不够,于是,可在计划中增加一个任务“客户交流”,并在计划执行中通过收集的度量数据确认需求缺陷是否回到

9.5±7.5 控制范围之内,否则就要采取新的纠正措施.

增加的任务既可以属于 OSP 中的某个活动,也可能不属于 OSP 中的任何一个活动.对于后一种情况,我们称其为项目的最佳实践(best practice,简称 BP).

通过对现有计划中任务的修改,也能够纠正项目执行过程中出现的偏差.例如,某项目控制基准中的进度控制阈值为延期不能超过ΔS,而项目度量数据发现$\Delta S'>\Delta S$,则通过对进度影响较大任务的修改,可以使得项目进度满足要求.下面,我们以关键路径分析法(critical path method,简称 CPM)为例,介绍如何通过修改任务实现对进度的控制.

在项目总存在关键一条关键路径(CP).$CP=\langle Task_{first},task_i,task_{i+1},\ldots,Task_{last}\rangle$,$Task_{first}$ 为项目的第 1 个任务,$Task_{last}$ 为项目的最后一个任务,且满足以下两个条件:

- 根据定义 5,$\forall Task_i\in CP$:($Task_{i+1}.Starttime=Task_i.Endtime$);
- $\Sigma(Task_i.EndTime-Task_{i-1}.Starttime)Task_{last}.Endtime-Task_{first}.StartTime$.

若通过分析在 t 时刻的任务完成情况的报告发现 CP 上的 $Task_i$ 延期$\Delta S'$,则可以考虑对关键路径上的 $Task_{i+1}$ 进行修改,从而满足进度控制要求.假定把 $Task_{i+1}$ 分解为进度等长的两个任务 $Task'_{i+1}$, $Task''_{i+1}$,只需要

$$(Task'_{i+1}.Endtime-Task'_{i+1}.Starttime)+(\Delta S'-\Delta S)=(Task_i.endTime-Task_{i-1}.StartTime).$$

类似地,通过删除任务也可以实现对项目执行过程的控制.与增加任务类似,如果任务删除后导致组织标准过程的活动从计划中删除,只要达到控制项目过程的目的,也可作为项目的最佳实践.我们将最佳实践放入一个集合 BP 中,$BP=\{Task_1,Task_2,\ldots,Task_n\}$.

2.3.2 对软件过程的改进

软件过程改进的目的是提升组织软件过程的能力,以便满足客户和软件组织自身发展的需要.但如果定义的软件过程不符合软件组织的实际情况,就谈不上对软件过程能力的改进.因此,本文对软件过程的改进涉及两个方面:(1) 软件过程的适应性;(2) 软件过程能力.软件过程改进的结果主要反映在对组织标准软件过程 OSP 中的活动、指导软件过程使用的裁剪指南和指导过程活动执行的指南/规范的重新修订上.

(1) 对软件过程的适应性改进

如前所述,每个项目在使用软件组织标准过程时,可根据项目的需要,选择软件过程中的活动在项目中执行.由算法 1,我们得到一组经过裁剪和为在项目中使用的活动集合 $P_{tg}=\{\ldots,AC''_{ij},\ldots,AC''_{mn},\ldots\}$,由于这些活动没有在项目中使用,或经过调整才能使用,说明这些活动对该项目的适用性不好.通过基于考察其他项目对过程的裁剪情况,得到 $P_{tg1},P_{tg2},\ldots,P_{tgn}$.令 $P=P_{tg1}\cap P_{tg2}\cap\ldots\cap P_{tgn}$,若 $P!=\varnothing$,则说明 P 中的活动适应性不好且带有普遍性,可以考虑加以改进,即改进组织 OSP 中的相应活动.

通过上一节讨论可知,各项目组在调整计划中的任务纠正软件过程出现的偏差时都会积累有些最佳实践 BP.考察各个项目组,得到 $BP=BP_1\cup BP_2\cup\ldots\cup BP_n$,$BP$ 中的任务可以考虑转化为 OSP 中的活动,从而可以在软件组织的项目中推广.由于这些活动来自项目的成功实践,因而具有较好的适用性.

另外,当外部环境发生变化时,也要考虑改进软件过程的适应性.其方式依然是通过调整 PA 以及 PA 中的活动 AC 来实现的.例如某公司发生业务转型,由应用产品开发转为外包型企业,那么,该企业过程中与需求开发和需求管理相关的 PA 及其活动 AC 则需要调整,以适应项目实际需要.

与此同时,软件组织标准过程定义,即 OSP 中的活动进行修改后,相应的指导软件过程使用的裁剪指南、指南/规范也应作出相应的调整.

(2) 对软件过程能力的改进

根据 Florac 等人的定义[21],软件过程的能力可以表示为软件过程的特征值及其允许的偏差范围.因此,提高软件过程能力就表现在对软件过程特征值的提高/降低和偏差范围的缩小上.根据定义 2,即 PPB 中特征值 V 和 m 的变化.但仅仅改变过程性能基线的值并不能提高软件过程能力,相关的软件过程才是真正需要改进的对象.

与对软件过程进行控制相类似,对软件过程的改进仍可使用质量管理的 7 种工具对过程度量数据进行分析,从而为软件过程的改进提供依据.但二者也有明显的不同:首先,为了对软件过程进行改进,利用质量工具对

数据进行分析的目的是找出影响软件过程性能的普遍性问题,而如果仅仅需要对软件过程进行控制,只要找出原因就可以;其次,修改的对象不同.对软件过程的控制可以通过修改项目计划来实现,而改进则主要是修改组织的软件过程,并要验证修改后的软件过程能否达到提升软件过程能力的目的.因此,软件过程改进应采用如下的步骤:

a) 利用质量管理的 7 种工具,对软件组织的各个项目过程的度量数据进行分析,识别出影响软件过程性能的普遍存在的原因;

b) 根据识别的原因,临时修改 OSP 中活动和设定新的 PPB 值;

c) 选择试点项目,并在该项目计划中使用新确定的 OSP 和 PPB;

d) 如果该项目的执行性能满足新的 PPB 要求,则正式修改 OSP 和 PPB,否则转到步骤 a).

另外,与提高软件过程的适应性一样,对于那些经过裁剪后被项目使用,或项目执行过程中被证明行之有效的最佳实践,如果不仅能够纠正软件过程出现的偏差,而且经过纠正的软件过程性能还具有更小的偏差范围、更高/更低的过程特征值,那么这些活动自然也是修改 OSP 的重要依据.

综上所述,基于 PDCA 的软件过程与控制模型通过面向过程的计划方法,并基于对度量数据的分析,可使得:

- 过程中的一系列活动通过计划变成了相关人员的日常工作(任务),使得项目相关人员“置身”于过程中,因而能够确保过程得以执行,有效避免管理与开发脱节、软件过程实施流于形式的问题;
- 软件过程在计划时可裁剪、在执行中可调整,加之强调了改进软件过程的适用性,因而能够有效避免软件过程繁琐僵化的问题,有助于提升软件过程的使用效果;
- 在软件过程执行过程中,数据能够得到及时收集和分析,有利于及时发现问题,从而使得对软件的执行过程和对组织软件过程的改进更客观,也提高了对软件过程控制与改进的效率.

3 SoftPM 介绍

根据“基于 PDCA 的软件过程改进控制模型”开发的软件质量管理平台 SoftPM 包含 3 个主要功能模块:项目管理/质量管理(PM)、软件过程管理(SPA)和软件度量(MT).其体系结构如图 3 所示.

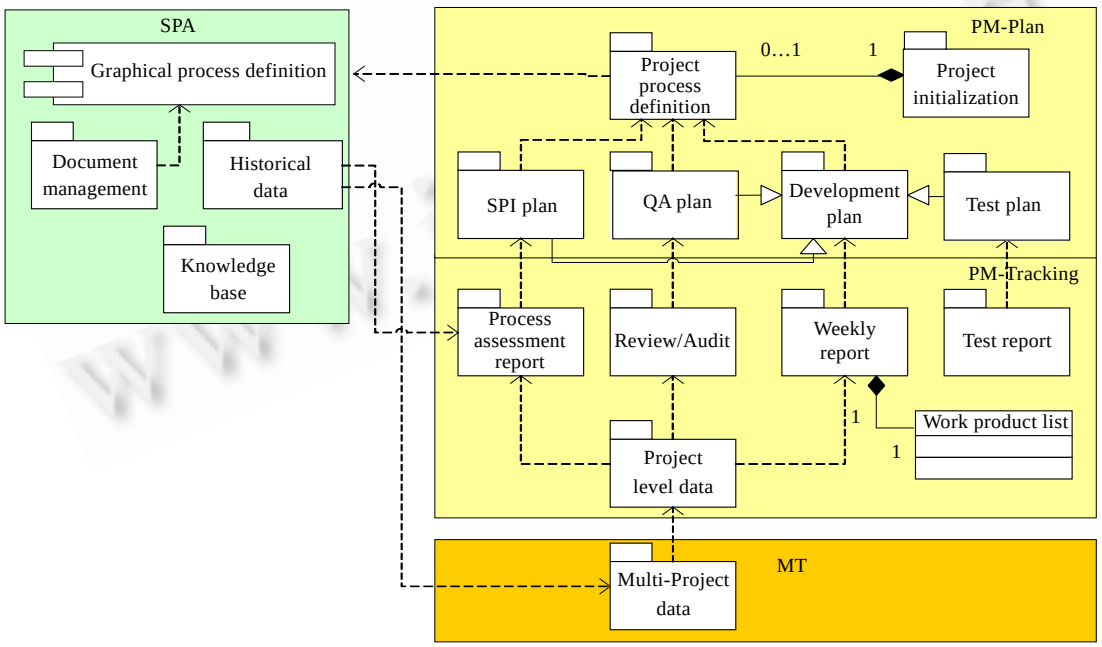

Fig.3 SoftPM architechture

图 3 SoftPM 体系结构

各个模块的主要功能如下:

- SPA(software process asset,软件过程资产)为软件过程管理模块,用于定义和维护软件组织标准过程 OSP 和相关过程资产,并围绕“成本、进度和质量”3 个要素定义了一组基础过程度量指标体系.OSP 可直接导入 PM-计划中,完成面向过程的工作拆分,同时也确定了对活动的度量指标.
- PM-计划和 PM 跟踪(数据采集)两个子系统构成软件项目管理模块,主要功能是制订和修改项目开发计划、过程改进计划、质量计划等,分配任务,通过各种任务报告跟踪项目进展,收集度量数据.
- MT 是软件度量分析模块,主要功能是对项目层采集的数据进行分析,并根据 7 种质量管理工具的要求,提供不同视图展示分析结果.而且,这些视图还根据系统使用者的角色不同,分不同的组合视图显示,以减少决策时的冗余信息.

项目计划的制订和修改均采用 Gantte 图表示形式,如图 4 中的每项任务均由 SPA 中的 OSP 导入,项目经理经过进一步细化后分配给相关人员.相关人员提交任务报告后(包括对任务的基础度量数据),系统进行关于进度、工作量、质量方面的分析;图 5 为挣值分析图,用以判断项目进展与计划的符合性;图 6 为缺陷密度的软件过程控制图(基于过程性能基线),用以判定软件过程是否存在异常,即是否超过了过程性能基线的要求;图 7 是给高层决策者提供的分析视图,高层可以明了地掌握组织内各个项目的整体进展情况.

Fig.4 Project plan Gantte chart

图 4 项目计划 Gantte 图

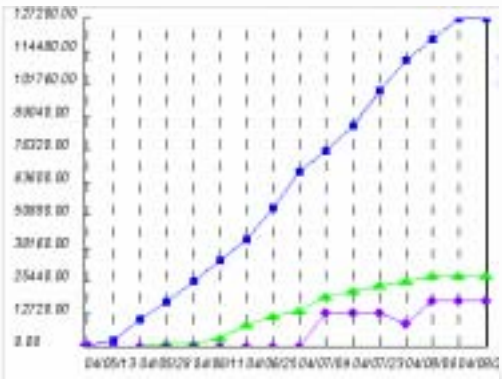

Fig.5 EV chart

图 5 挣值分析图

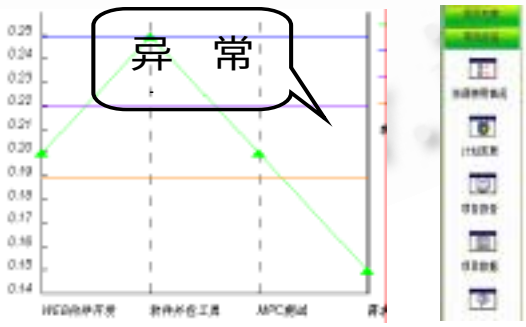

Fig.6 Control chart

图 6 控制图

Fig.7 Multi project progress

图 7 多项目进展

系统采用基于 J2EE 的 B/S 结构,还支持任务多层分解和协同编辑,可有效支持对异地大型开发项目的协同开发管理.

另外,通过配置资源文件,系统实现了对多语言的支持.

3.1 系统特点

- 质量平台提供的软件过程资产管理功能,为企业建立组织标准过程提供了良好的支持.其中图形化过程定义窗口方便了 SEPG 人员制定公司的标准过程,也易于被公司开发人员接受、理解和使用,使过程建立、维护和使用组织标准过程的难度大为降低.
- SoftPM 实现了目前国际流行的 Gantt 图项目计划方法,使得项目经理很方便地利用 Gantt 图进行按过程进行工作拆分、配置资源、指定和检查工作产品、安排进度,项目管理的效率有了极大的提高.
- SoftPM 提供挣值分析、资源利用等度量数据、问题/风险处理等功能,可以实时了解项目的进度和成本,及时发现问题并采取纠正措施,为高层经理和项目管理人员提供了非常有效的项目控制手段.

目前,SoftPM 已经在北京、西安、成都、江苏、珠海、长沙、上海、广州共 8 个国家软件产业基地得到使用或试用,覆盖率达 11 个国家软件产业基地的 73%;在广东、北京、上海、四川、西安、沈阳、河南、昆明这 8 个国家科技部批准的 863 软件专业孵化器使用,覆盖率 100%.

SoftPM V0.5,V1.0,V2.0,V2.5 和 V2.6 等各种版本用户超过 100 家.为北京等国家软件产业基地和 863 专业孵化器提供 12 000 多个授权用户数,为软件企业提供的授权用户数近 4 000 个.

从企业用户的反映来看,本系统支持基于 CMM/CMMI 软件过程执行.某公司在 SoftPM 支持下,2004 年底通过 CMMI5 评估.通过使用 SoftPM 和相关服务,软件开发规范化程度有了显著的提高,降低了管理成本,提高了管理效率,增强了公司在市场上的竞争力.有些企业争取的项目数量比以往提高 40%,项目周期平均缩短 30%,直接为公司创造了较大的经济效益.

4 结 论

基于 PDCA 的软件过程控制和改进模型采用面向过程的项目计划方法,将软件组织的标准过程转换成软件项目相关人员的任务,能够有效保证过程得以执行.通过对度量数据的分析,能够客观地决定如何对项目执行过程进行控制以及对组织标准过程进行改进.同时,通过对项目计划 s 中任务的调整和对组织标准过程得到改进,还能够满足软件过程适应外部环境动态变化的要求.

基于该模型开发的 SoftPM 平台目前在国内已经得到了比较广泛的应用.企业用户普遍认为 SoftPM 平台对软件企业提高管理效率、降低管理成本有很大帮助.

该成果的应用能够克服目前在我国实施 CMM/CMMI 中存在的流程繁琐僵化、实施成本高的问题,推进 CMM/CMMI 在我国软件企业中实施,从而促进软件企业整体提高管理水平.

下一步的研究重点是进一步加强软件质量管理平台与工程过程支持工具的数据集成,从而为软件企业的产品开发与管理提供一个更为完善的解决方案.

References:

[1] Humphrey W. Managing the Software Process. New York: Adison-Wesley, 1989. 5−10.

[2] Jalote P. Moving from ISO9000 to higher levels of the CMM. In: Proc. of the 22nd Int'l Conf. on Software Engineering. Limerick, ACM Press, 2000. 823.

[3] Paulk MC, Weber CV, Garcia SM, Chrissis MB, Bush M. Key practices of the capability maturity model^{SM}, Version 1.1. Technical Report, No.CMU/SEI-93-TR-025, Pittsburg: Software Engineering Institute, Carnegie Mellon University, 1993.

[4] SEI. Process Maturity profile of the software community. Pittsburg: Software Engineering Institute, 2004.

[5] CMMI Product Team. Capability maturity model$^{®}$ integration (CMMI^{SM}), Version 1.1. Pitsburg, Software Engineering Institute, 2001.

[6] Osterweil L. Software process are software too: Revisited. In: Proc. of the 19th Int'l Conf. on Software Engineering. New York: ACM Press, 1997. 540−548.

[7] Curtis B, Kellner MI, Over J. Process modeling. Communications of the ACM, 1992,135(9):75−90.

[8] Ambriola V, Conradi R, Fuggetta A. Assessing process-centered software engineering environment. ACM Trans. on Software Engineering and Methodology, 1997,16(7):283−328.

[9] Zhao X, Chan K, Li M. Applying Agent technology to software process modeling and process-centered software engineering environment. In: Proc. of the 20th Annual ACM Symp. on Applied Computing (SAC 2005). New York: ACM Press, 2005. 13−17.

[10] PMI. A Guide to Project Management Body of Knowledge. 2000 ed., Newtown Square: Project Management Institute, Inc., 2000.

[11] Harada A, Awane S, Inoya Y, Ohno O, Matsushita M, Kusumoto S, Inoue K. Project management system based on work-breakdown-structure process model. In: Li M, Boehm B, Osterweil LJ, eds. Proc. of the Software Process Workshop. Beijing: Springer-Verlag, 2005.

[12] Abrahamsson P. Commitment development in software process improvement: Critical misconceptions. In: Proc. of the 23rd Int'l Conf. on Software Engineering. Hawaii: IEEE press, 2001.

[13] van Soligen R, Berghout E. The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development. London: Mcgraw-Hill, 1999. 10−22.

[14] Wang Q, Li MS, Liu X. An active measurement model for software process control and improvement. Journal of Software, 2005,16(3):407−418 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/407.htm

[15] Diaz M, Sligo J. How software process improvement helped Motorola. IEEE Software, 1997,14(5):75−77.

[16] Shumate K, Snyder T. Software reporting: Management, measurement, and process improvement. In: Proc. of the Conf. on TRI-Ada'94. New York: ACM Press, 1994. 41−45.

[17] Vu JD. Process improvement in retrospective: Lessons learned from software projects. In: Proc. of the SEPG Conf. Seattle: SEI, 2005.

[18] Sugahara K, Ogasawara H, Aoyama T, Higashi T. Status of SPI activities in Japanese software—A view from JASPIC. In: Li M, Boehm B, Osterweil LJ, eds. SPW 2005. LNCS 3840, Heidelberg: Springer-Verlag, 2005. 407−420.

[19] QAI. The India software success story. 2002. http://www.qaiindia.com/Misc/archives/sepg_2002.pdf

[20] Wu ZC, Christensen D, Li MS, Wang Q. A survey of CMM/CMMI implementation in China. In: Li M, Boehm B, Osterweil LJ, eds. Proc. of the Software Process Workshop 2005. Beijing: Springer-Verlag, 2005. 507−520.

[21] Florac WA, Park RE, Carleton AD. Practical software measurement: Measuring for process management and improvement. Technical Report, CMU/SEI-97-HB-003, Pittsburg: SEI, 1997.

附中文参考文献:

[14] 王青,李明树,刘霞.一种支持软件过程控制和改进的主动度量模型.软件学报,2005,16(3):407−418. http://www.jos.org.cn/1000-9825/16/407.htm

武占春(1965－),男,吉林德惠人,博士,主要研究领域为软件质量管理.

王青(1964－),女,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为软件工程,软件质量管理.

李明树(1966－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为软件工程,实时系统.