CMM 过程支持系统中对过程改变多策略支持的研究

胡 昊, 林向宇, 杨 玫, 吕 建
(1. 南京大学计算机软件新技术国家重点实验室, 江苏南京 210093; 2. 南京大学计算机软件研究所, 江苏南京 210093)

摘 要: 企业在实施 CMM 时, 过程改进这样的宏观目标是通过从微观上成功地完成项目来实现的. 因此, 自动实施 CMM 的过程支持系统应该具有灵活的支持过程动态改变的能力, 以利于适应多变的项目环境, 而过程改变的情况比较复杂, 单一策略的采用既增加了模型的复杂性, 又不利于系统充分利用已有的底层支持技术. 本文介绍了一种多策略过程动态改变支持机制. 该机制可用于基于 CMM 的过程支持系统中, 有效提高基于 CMM 的过程管理系统在过程改变支撑方面的灵活性和实用性.

关键词: 过程动态改变; 过程支持系统; 能力成熟度模型; 过程模型; 过程改进.
中图分类号: TP311 文献标识码: A 文章编号: 0372-2112 (2003) 12A-2087-04

The Research on Multiple Tactics to Support Process Change in PSS for CMM

HU Hao, LIN Xiang-yu, YANG Mei, LU Jian
(1. *State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu 210093, China;*
2. *Institute of Computer Software, Nanjing University, Nanjing, Jiangsu 210093, China*)

Abstract: In application of CMM, the general purpose of process improvement is fulfilled by successful completion of various projects within an enterprise. As an automated platform, the process support system (PSS) should have the ability to support dynamic change of process, in order to respond to the environments reactively. However, process change is so complex that single tactic adopted in PSS not only increases the process model's complexity but also weakens the reuse of the underlying technologies which have already been used in PSS. This paper proposes a multi-tactic support mechanism for dynamic change of process. This mechanism, used in PSS for CMM, increases the flexibility and usability of similar system to support process change.

Key words: process dynamic change; process support system; CMM; process model; process improvement

1 引言

实践证明, 实施 CMM 能有效提高软件企业的生产效率及软件的质量[1]. 实施 CMM 的高效方法是对软件过程进行自动化[2], 自动化的途径之一是使用"过程支持系统" (Process-support system). CPMS(CMM-based process management system) 是由南京大学软件新技术国家重点实验室开发的一个基于 CMM 的过程支持系统, 它能够支持符合 CMM 规范的软件过程.

一般情况下, 一个企业能否成功的实践 CMM, 一方面是从宏观的角度看该企业是否通过过程改进提高了能力成熟度级别, 另一方面更要从微观的角度看企业能否去成功的完成一个个项目. 而从实践 CMM 的最终目的来看, 后一方面更是前一方面目的的体现. CMM 强调成熟度高的软件企业其软件过程应该具有稳定, 有规律, 可重复, 可量化, 可预测的特点[3], 然而过分僵化的理解 CMM 的这个含义容易导致忽视单个项目的特殊性. 即使是一个 CMM 实践相当成熟的企业, 也会由于环境、人员、技术、资金等诸多因素的不确定性, 使其管理下的项目具有复杂性和多样性. 因此, 帮助企业自动化实施 CMM 的过程支持系统必须具有灵活的适应过程动态改变的能力.

传统的过程支持系统大多采用"过程编程"的方法, 该方法通常采用"预设的过程控制"手段, 其含义是先建立过程模型, 然后严格地对过程模型实施解释和控制. 这样的过程支持系统对复杂和多变的软件项目开发环境缺乏适应性, 无法灵活有效地支持频繁变化的项目. 因此, 对过程动态改变支持的研究成为过程支持系统研究中的重点[4].

已经有许多对过程动态改变支持的研究, 然而大多数研究集中于对过程模型和支持系统的改进上, 希望通过增加模型的灵活性来达到对系统灵活性的支持. 但是在实际的 CMM 的实施中, 过程动态改变的情况比较复杂, 一个完全通用的模型未必能够覆盖到所有的情况, 而且单纯的改进过程模型也增加了模型的复杂性, 因此 CPMS 采用了多个策略来解决过程动态变化的支持问题, 这样可以较灵活的提供多种过程动态改变的手段.

本文针对企业实施 CMM 时对过程动态改变的需求, 分别

收稿日期: 2003-09-30; 修回日期: 2003-12-16
基金项目: 国家 863 高科技计划(No. 2001AA113090)

有针对性地在 CPMS 系统中设计并实现了三种过程动态调整的策略.

2 CMM 实施的新型理念模型

CMM 传统认为,一个具有稳定,有规律,可重复,可量化,可预测的软件过程对提高企业的软件质量和软件生产率有很大帮助,这样的理念使得传统的过程支持系统普遍采用"预设的过程控制"手段进行"过程编程",对过程变化的支持缺乏灵活性.然而现在这样的观点遭到了来自轻量型软件方法学的挑战[5].轻量型软件方法学强调软件过程是一个不断的演化过程,必须随时对环境的变化做出调整,这恰恰与 CMM 所强调的稳定,有规律的特点相矛盾,因此关于软件过程是有规律(discipline)还是敏捷的(agile)一直都是学术界争论的焦点[6].

如果将轻量型方法所提倡的过程动态变化的特点考虑在 CMM 中,就可以建立一种新型的 CMM 实施的理念模型(如图 1),在该模型中,我们认为 CMM 在一个企业中应当体现为宏观上的过程改进和微观上的项目执行.从组织的高层视图看,企业有一个宏观上稳定的过程改进目标,要想实现目标,企业必须从微观上成功地完成一个个项目.然而由于环境、人员、技术、资金等诸多因素的不确定性,使得企业所面对的项目具有多样性.早期项目所产生的过程经验(组织级软件过程)如果不经过调整演化,就会变得僵化,很难适应项目所面对的多变的环境要求.过程支持系统应该从微观上支持企业成功的完成项目,这就要求象 CPMS 这样的系统应该具有灵活的机制来适应不断变化的过程改变的要求.综合来说,这个新型模型有这样的理念:通过微观上频繁、实时地调整、变化、适应可以建立宏观上的稳定和规律.这样的观点在一个改进的 CMM 模型 CMM-Taiji[7]中也有很好的印证.

从上面的论述可知,对过程动态改变的研究已经成为设计过程支持系统的重要因素,在总结归纳 CMM 实施时实际的过程改变情况,可以将过程的动态改变分为如下三种类型(如图 1):(1)Trivial 型.过程活动之间的依赖关系没有变化,而与活动相关的产品数据和执行成员却经常改变.比如一个变更控制过程,要经历变更请求、变更实现、测试、评审、发布这五个阶段,其过程活动相对固定,而每次的参加人员和控制文档则有可能不同.(2)Limited 型.过程活动之间的依赖关系发生变化,但是其变化被局限于一个局部的子过程内.如某个过程中规定测试需要进行三次,但是发现其测试结果未能达到预期的检错率,从而又增加了一次测试;又如在某个开发时刻由于市场的变化而要求交付日期提前,则此时需要减少某些活动等.(3)Overall 型.过程活动之间的依赖关系发生变化,而且其变化改变了整个过程.在项目执行中由于技术,资金,时间的制约,项目的执行过程被完全改变.

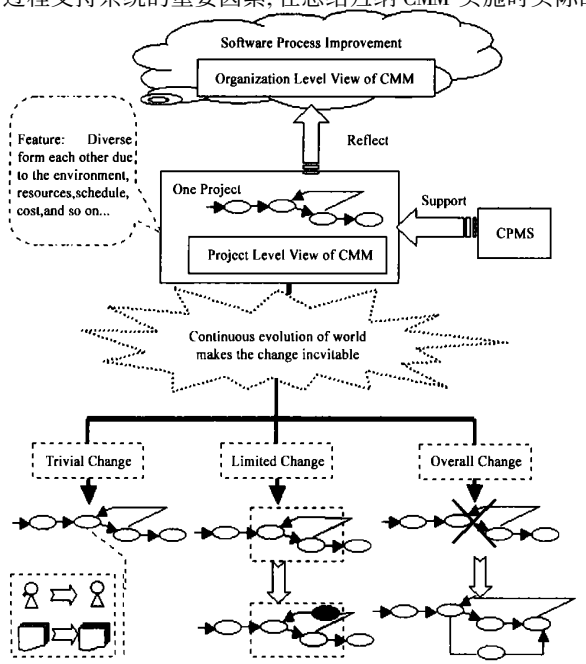

图 1 CMM 实施的理念模型

3 CPMS 系统简介

CPMS 是一个分布式的过程支持系统,其主要由过程定义部分和过程解释执行部分组成.过程定义部分包括过程定义工具、角色定义工具和文档定义工具,它们分别用来建立 CPMS 的活动流模型、组织模型和数据流模型.CPMS 使用 SPDL(Software process description language)为上述三种模型建模,SPDL 是一个 XML 兼容的模型语言.解释执行部分的核心是过程控制引擎.过程引擎解释执行由 SPDL 描述的过程,分派任务给各个用户,并通过 Web 界面使用户执行任务并和过程引擎进行交互.

CPMS 的过程引擎支持三种基本的控制结构,分别是顺序、选择和并发.顺序和选择结构与程序设计语言中的顺序和多分支结构相同;并发结构与选择结构类似,但它不做选择,而是并行执行多个分支.

这些基本的控制结构对于灵活地支持复杂的过程仍然存在结构上的不足,因此过程引擎还支持一类称为"例程"的特殊子过程.引擎支持例程的方法类似于程序设计语言中的异常处理.

4 CPMS 对过程改变的多策略支持

4.1 多策略解决过程动态改变的原因

已经有许多对过程动态改变支持的研究,如在文献[8]中采用的基于规则的方法,在文献[9]中采用的自省的解决方法,在文献[10]中提出了采用多 Agent 系统的方法,这些研究方法大多集中于对过程模型和支持系统的改进上,希望通过增加模型的灵活性来达到对系统灵活性的支持.但是在实际的 CMM 的实施中,过程动态改变的情况比较复杂,一个完全通用的模型未必能够覆盖到所有的情况,而且单纯的改进过程模型也增加了模型的复杂性,因此 CPMS 采用了多个策略来解决过程动态变化的支持问题,这主要是有如下两点考虑:

(1)简化 CPMS 的过程模型.过程模型应该在更高的层次上描述过程元素的关系,简化的过程模型对语法的验证和语义的描述都有帮助.同时,运行支撑与过程模型是紧密相关的,复杂的过程模型必然带来过程引擎设计的复杂性.

(2)充分利用 CPMS 已有支撑技术.这样有两点好处,一是可以重复利用已有的成熟技术,避免新技术的引入所带来

的系统不稳定的风险,二是降低了 CPMS 系统实现的复杂性.

4.2 过程改变多策略支持的设计

为了简化过程模型和充分利用已有技术,CPMS 系统采用了三种策略来解决过程动态改变的三种类型:(1) 成员—文档后绑定策略;(2) 过程活动局部黑盒策略;(3) 过程实例整体替换策略.

4.2.1 成员– 文档后绑定策略　该策略主要解决过程动态改变的 Trivial 型,通过为过程活动和活动的其他属性(如 Agent, Resources) 分别建立模型,可以后绑定活动所需的资源和成员. 在 CPMS 中,除了过程的活动流模型外,还有组织模型和数据流模型. 每个活动通过角色、输入、输出文档与组织模型和数据流模型建立联系,而组织模型中的"角色-成员"列表帮助过程引擎在运行时动态确定活动的执行成员. 相同的情况在数据流模型中也有所体现,"文档模板– 实例列表"也可以帮助过程引擎在运行时动态确定活动所需访问的输入输出文档.

4.2.2 过程活动局部黑盒策略　为解决过程动态改变的 Limited 型,CPMS 系统借鉴了文献[11]中有关黑盒的方案,对过程中由于局部动态变化而不能确定的子过程用一种新的节点类型"黑盒"来表示(如图 2),黑盒可以替换掉过程中某个结构化的子过程,并由过程引擎在运行时动态确定黑盒的语义,从而执行替代黑盒的过程片段.

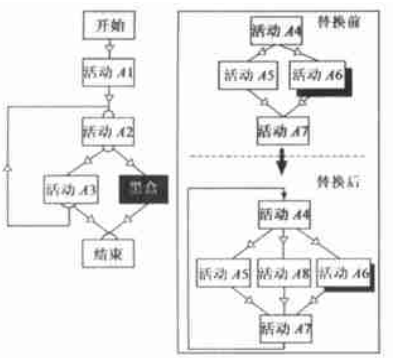

图2　用节点类型"黑盒"表示的子过程

在该策略的系统实现中,过程引擎利用 CPMS 中的"例程"机制对黑盒进行替换."例程"可以使"黑盒"具体化,它借鉴了异常处理机制,通过将某些子过程描述成例程,可以在引擎运行时动态加载这些子过程,加载的条件可以由用户决定或由过程引擎根据条件判断. 这样,通过在执行黑盒前,将黑盒具体化为某个例程,就可以实现过程引擎动态改变黑盒所描述的过程片段.

4.2.3 过程实例整体替换策略　该策略用来解决过程动态改变的 Overall 型,即在过程的执行中对过程的整体结构进行改变. 通常对过程的全局结构进行动态修改是很复杂的,其中主要解决过程模型和过程实例之间的一致性问题,一般情况下,对全局的调整有两种方案:

(1) 其一是将过程模型修改,然后启动引擎从头开始解释执行,这样过程模型的修改和过程实例的修改是同步进行的.

(2) 另一种情况是过程模型的修改不直接干预过程实例的运行,而是指导过程实例的运行,即过程实例的修改延迟于模型的修改,如在文献[12]中所给出的方法,即容忍过程实例与过程模型的偏差,在适当时机调整过程模型与过程实例之间的不一致.

CPMS 采用了前一种策略,但是 CPMS 不用重新启动过程引擎,过程引擎将在过程模型修改后得到原语通知,执行新的过程实例.

该策略的运行方式如下,当过程模型被修改后,系统向过程引擎送达一条原语,RESTART(returnpoint),引擎将该返工点"returnpoint"与过程实例现在执行的活动 ID 做比较,根据比较的结果产生相应的处理:第一种情况是返工点在过程实例的活动 ID 之后,此时过程引擎无需做任何操作,只要按照正常的过程活动执行即可;第二种情况是返工点在过程实例的活动 ID 之前,即过程实例的运行活动已经超过了返工点,此时表示项目过程有一部分需要重新执行,需要回退到返工点,重新从该点开始执行新的过程模型定义. 在回退的过程中要保证回退到返工点的活动状态与正常执行原有过程模型时到达该点的活动状态能够保持一致,CPMS 的过程引擎提供了机制保证一致性,这样的机制称为"UnDoing"机制.

"UnDoing"机制的实现方式是将过程实例中活动的正常状态设置成"无效",如图 3 所示,一般活动在正常情况下有"正处理、待启动和已完成"三种状态,当实现"UnDoing"时,过程引擎将把已经执行的过程实例中那些为"正处理"和"待启动"状态的活动都设置为"无效"状态,过程引擎对"无效"状态的活动将不产生任何执行动作. 过程引擎将重新对新的过程模型开始执行.

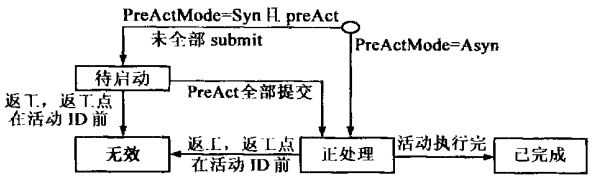

图 3　活动的状态转换图

4.3 过程改变多策略支持的实现

在 CPMS 中,"成员—文档后绑定"策略是通过 SPDL 为活动流模型、组织模型和数据流模型分别建模来实现的. 以成员后绑定为例(如图 4 所示),活动流模型由"活动"数据表构成,由过程定义工具负责实例化;而组织模型由"角色"、"角色成员"及"成员"三张数据表构成,由角色定义工具负责实例化并可在过程执行期间动态更改. 成员后绑定的执行过程如下:过程引擎(如图 5) 解释流程时先只参考活动流模型,仅在活动开始执行需要确认成员时才由 Handler 模块读取组织模型以获取成

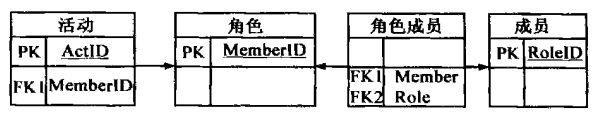

图 4　成员后绑定模型

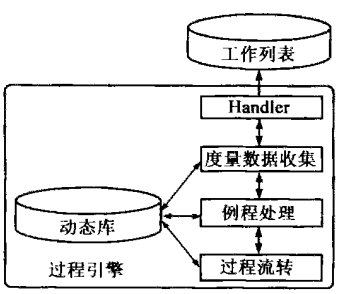

图 5　成员后绑定的执行过程引擎

员,从而实现成员的后绑定. 通过类似的方式,CPMS 也实现了资源的后绑定.

“过程活动局部黑盒”策略的实现主要依赖于过程引擎中的例程处理模块(如图 5),图 6 是当主过程遇到黑盒时启动例程运行的伪代码. 当过程活动为黑盒活动时,如果该黑盒活动已指定了相应的例程,则过程引擎将启动该例程运行,经过“初始化例程”,“挂起主过程”,“运行例程”,和“重新启动主过程”四个阶段执行例程. 这样,通过修改黑盒所对应的例程,即实现了过程局部黑盒的变更.

```
// Main process:
do {
    if(activity= = BLACKBOX)
    {
        if(Routine for BLACKBOX= = ASSIGNED)
            Start Routine for BLACKBOX
        else
            // Notify to Process Manager
    }
else
    do activity;
} while((activity= getNextActivity()) ! = END);
/* * * * * * * * * * * * * * * * * * * * *
// Routine process:
Initialize; // set doc, member and activity ready for running
Block Main;
Run Routine;
Restart Main;
```

图 6 主过程遇到黑盒时启动例程运行的伪代码

“过程实例整体替换”策略主要是通过过程引擎修改动态库中活动的执行状态来完成的. 当过程引擎接收到 RESTART 原语,且返工点与过程实例的活动 ID 的比较结果为上文中提到的第二种情况时,过程引擎将从返工点开始,顺次将执行流程中返工点后的活动状态修改为“无效”,并从返工点开始执行新定义的过程模型.

5 总结

过程支持系统要想适应企业灵活多变的项目环境,就必须具有支持过程动态改变的能力. PIE 平台[13]就是一个能够支持过程动态改变的典型系统,其建立了多个支持演化的组件和一个支持组件交互的通信中间件,通过过程模型的“分片”技术,可以动态的改变运行时的过程实例.

大多数此类研究所采用的方法都集中于增加过程模型的语义或者丰富底层的支撑技术,然而这样的系统往往会陷入设计野心过大的灾难,系统由于过程模型过于复杂,或者底层支撑技术过多而难于设计、集成,缺乏实际的使用能力.

相比较而言,CPMS 的设计思想在于:不试图解决过程动态改变的所有情况,而是针对 CMM 在实践时所遇到的典型过程改变类型来设计解决过程动态改变的策略,通过多策略的引入来简化系统的模型设计和提高底层支撑技术的利用,以增加系统的灵活性和实用性. 这个设计思想的优点已经通过系统实现并在金蝶等公司内的试用而得到了验证.

参考文献:

[1] Software Engineering Institute, Carnegie Mellon University. Capability Maturity Model: The Guidelines for Improving the Software Process [M]. Boston, USA: Addison-Wesley, 1995.

[2] Watts S Humphrey. Managing the Software Process[M]. Boston, USA: Addison-Wesley, 1989.

[3] Mark C Paulk, et al. Capability maturity model, Version 1.1[J]. IEEE Software, 1993, 10(4): 18-27.

[4] G Cugola, C Ghezzi. Software processes: A retrospective and a path to the future[J]. Software Process Improvement and Practice, 1998, 4(3): 101- 123.

[5] Kent Beck. Embracing change with extreme programming[J]. IEEE Computer, 1999, 32(10): 70- 77.

[6] Kent Beck, et al. Agility through discipline: A debate[J]. IEEE Computer, 2003, 36(6): 44- 46.

[7] Zhou Zhiying. CMM in uncertain environments[J]. Communications of the ACM, 2003, 46(8): 115- 119.

[8] Gerti Kappel, et al. A framework for workflow management systems based on objects, rules and roles[J]. ACM Computing Surveys (CSUR), 2000, 32(1es): 27.

[9] D Edmond, et al. A reflective infrastructure for workflow adaptability [J]. Data and Knowledge Engineering, 2000, 34(3): 271- 304.

[10] Paul A Buhler, et al. Adaptive workflow= Web services+ agents[A]. Liang-Jie Zhang, et al. Proceedings of the International Conference on Web Services, ICWS 2003 [C]. Las Vegas, Nevada, USA: CSREA Press, 2003. 131- 137.

[11] 孙瑞志,史美林. 支持工作流动态变化的过程元模型[J]. 软件学报, 2003, 14(1): 62- 67.

[12] Gianpaolo Cugola. Tolerating deviations in process support systems via flexible enactment of process models[J]. IEEE Transactions on Software Engineering, 1998, 24(11): 982- 1001.

[13] Pierre-Yves Cunin, et al. The PIE methodology-Concept and application[A]. Vincenzo Ambriola et al. LNCS 2077[C]. Witten, Germany: Springer Verlag, 2001. 3- 26.

作者简介:

胡 昊 男,1975 年 4 月生于江苏南京,2000 年毕业于东北大学,获硕士学位,现为南京大学计算机科学与技术系讲师,南京大学博士研究生,主要研究方向为软件过程,工作流技术,软件 Agent 技术.

林向宇 男,1979 年 9 月生于福建晋江,2001 年毕业于南京大学,获学士学位,现为南京大学计算机科学与技术系硕士研究生,主要研究方向为软件过程,工作流技术.