基于 SPEM 的 CMM 软件过程元模型*

李 娟[1,2+], 李明树[1,3], 武占春[1], 王 青[1]

[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)

[2](中国科学院 计算机科学重点实验室,北京 100080)

A SPEM-Based Software Process Metamodel for CMM

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

[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 of Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

+ Corresponding author: Phn: +86-10-82620803 ext 814, E-mail: lijuan@itechs.iscas.ac.cn, http://www.itechs.com.cn

Received 2004-05-31; Accepted 2004-10-09

Li J, Li MS, Wu ZC, Wang Q. A SPEM-based software process metamodel for CMM. *Journal of Software*, 2005,16(8):1366−1377. DOI: 10.1360/jos161366

Abstract: In applications of CMM (capability maturity model for software), it is a sticking point to implement enforceable software process models, which reflect the characteristics of organizations and their software processes, by transforming CMM software process model. Model driven architecture (MDA) supports model transformation and can be used in CMM practices, but the first step is to build a software process metamodel for CMM. This paper presents a software process metamodel for CMM based on SPEM(software process engineering metamodel), named SPM-CMM. SPM-CMM provides abstract syntaxes and rule semantics for CMM software process, and also supports of modeling of CMM integration with UML CASE tools.

Key words: CMM; software process metamodel; SPEM(software process engineering metamodel); MDA(model driven architecture); model transformation

摘 要: 软件企业在实施 CMM(capability maturity model for software)的过程中面临最主要的障碍是如何将 CMM 软件过程模型转换成可实施的、体现组织过程特征的 CMM 实施过程模型.可以利用模型驱动架构 MDA 来支持 CMM 模型转换,其首要问题是建立 CMM 软件过程元模型.通过分析 CMM 软件过程,给出了面向 CMM 的软件过程工程元模型 SPEM 的扩展策略,提出了一个基于 SPEM 的 CMM 软件过程元模型——SPM-CMM.该元模型既支持

* Supported by the National Natural Science Foundation of China under Grant No.60273026 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant Nos.2002AA116060, 2001AA113080 (国家高技术研究发展计划(863))

作者简介: 李娟(1977−),女,山东德州人,博士生,主要研究领域为需求工程,模型驱动架构,过程改进;李明树(1966−),博士,研究员,博士生导师,CCF 高级会员,主要研究领域为智能软件工程,实时系统;武占春(1965−),博士生,主要研究领域为软件过程改进;王青(1964−),女,博士,研究员,主要研究领域为软件质量管理,过程建模,知识管理,软件协同工作,实时系统开发方法.

CMM 软件过程的抽象语法和规则语义,也支持利用 UML CASE 工具操作 CMM 软件过程模型.
关键词: CMM;软件过程元模型;软件过程工程元模型;模型驱动架构;模型转换
中图法分类号: TP311 文献标识码: A

软件能力成熟度模型 CMM(capability maturity model for software)自问世以来,在软件过程改进方面取得了十分可观的成绩.CMM 的关键作用在于它提供了有效的模型来辅助企业评估和改进软件过程.但是,CMM 并没有给出实施软件过程改进活动的细节,从而限制了 CMM 在过程改进中更为有效的应用.Carnegie Mellon 大学的软件工程研究所 SEI(Software Engineering Institute)经过研究发现,多数企业希望在如何实施过程改进活动方面获得指导[1].一个企业要实施 CMM,就必须经过一个对 CMM 量体裁衣的过程[2],并且当企业的业务过程发生改变时,CMM 实施过程也要相应地改变.因为实施 CMM 需要大量人力、物力的支持[3],同时也要承担相应的风险,如何灵活地裁剪 CMM 是软件过程改进中的重要问题.Ginsberg 等人提出一种 CMM 的过程裁剪框架[4],其目的是结合组织需求,将 CMM 裁剪成组织标准软件过程,但由于该框架本身缺乏实用性,因而未得到广泛应用.另外,也有针对小型组织实施 CMM 的研究[5,6],但是这些方法均缺乏通用性.定义一种反映最佳实践的过程是进行灵活过程裁剪的有效方法[7],然而如何发现和判断问题域是需要解决的重要问题.

裁剪 CMM 的过程本质上是将 CMM 软件过程模型转换成可实施的、体现企业过程特征的 CMM 实施过程模型.模型驱动架构 MDA(model driven architecture)为解决模型转换问题开辟了新的思路[8].MDA 的基本思想是把业务逻辑从实现细节中抽象出来,将平台无关模型 PIM(platform independent model)转换成平台相关模型 PSM(platform specific model),从而达到 PIM 在不同平台上实现的目标.实施 CMM 时,企业本身是一个实施平台,CMM 软件过程模型则是与企业平台无关的 PIM 模型,实施 CMM 就是将这个 PIM 转换成满足不同企业平台要求的 PSM,即 CMM 实施过程模型.利用 MDA 框架所提供的模型转换设施,可以支持 CMM 软件过程模型向不同企业的 CMM 实施过程模型转换,从而辅助进行过程改进,提高实施 CMM 的效率,减少成本并降低风险.实现这种模型转换的首要问题就是如何在 MDA 框架下建立 CMM 软件过程模型.

软件过程模型描述了软件过程所涉及的活动、人员、资源等各种信息,是进行过程分析和实施的基础.目前已有多种软件过程建模方法被提出,例如基于 Agent 的过程建模[9]、基于统一建模语言 UML(unified modeling language)的过程建模[10]等.对 CMM 建模的主要方式是采用 UML[11].这种建模方法使用类图表示 CMM 中主要元素(关键过程域、目标等)之间的关系,虽然在一定程度上表达了 CMM 的内容,然而因为缺乏对元模型系统而完整的描述,导致对 CMM 特定过程语义的支持不足.

本文在扩展软件过程工程元模型 SPEM(software process engineering metamodel)[12]的基础上提出一个 CMM 软件过程元模型——SPM-CMM(software process metamodel for CMM).该元模型支持 MDA 框架,有利于模型的互操作以及获得 UML CASE(computed aided software engineering)工具支持,同时支持 CMM 特定过程语义,并利用 SPEM 丰富的过程元素,增强 CMM 软件过程模型的可操作性.

本文第 1 节说明 CMM 软件过程,并给出面向 CMM 的 SPEM 扩展机制.第 2 节论述 CMM 软件过程元模型 SPM-CMM,包括其抽象语法、规则语义和 UML Profile 机制.第 3 节介绍基于 SPM-CMM 元模型的原型工具 MDA-SPMT(MDA-Based software process model transformation)及其对 CMM 过程建模的支持.第 4 节总结全文并说明进一步的研究方向.

1 CMM 软件过程建模

1.1 CMM软件过程

CMM 是一个软件过程改进模型,由于未提供其规定的过程改进活动的具体实现方法,因此可操作性较弱.软件过程包括 3 类基本元素:组成过程的活动、约束活动进行的规则以及执行活动时需要使用的或者生产的对象,例如人力资源、信息、产品和工具等[13].在 CMM 的 5 个成熟度级别中,每一级都包含若干个关键过程域 KPA(key process area).KPA 定义了一系列的相关活动,通过执行这些活动来达到一定的目标[14].CMM 软件过程

是对 CMM 中定义的活动、规则、人员以及工作产品等信息进行组织和细化,以易于理解、过程化的方式进行表示,从而在一定程度上增强 CMM 的可操作性.

CMM 软件过程可以从“成熟度级别”和“KPA”两个层次来说明.从“成熟度级别”这一层来看,各 KPA 并非独立存在,相互之间存在依赖关系.图 1 以 CMM2 可定义级为例,说明需求管理 RM(requirements management)、软件项目计划 SPP(software project plan)、软件计划跟踪与评估 SPTO(software plan track oversight)、软件质量保证 SQA(software quality assurance)和软件配置管理 SCM(software configuration management)KPA 之间的依赖关系.RM 中确定需求之后,开始进行 SPP 的制定项目计划,该 KPA 应产生项目开发计划、SQA 计划和 SCM 计划.计划执行阶段,需要进行 SPTO 计划跟踪.同时,在开发阶段中产生的里程碑式的工作产品需纳入 SCM 配置管理,放入配置库中.作为支持活动的 SQA,需要定期进行审计和评审,以保证整个项目按照计划执行.从“KPA”这一层来看,各 KPA 中的内容也是符合过程特征的,因为其中包含一系列相关的活动、人员、目标、约束条件以及资源等.例如 RM 需求管理中,规定需求管理活动必须依据成文的组织方针进行,由软件工程组参与需求管理活动,还应当执行相应的验证和度量活动,并且要求组织提供支持需求管理活动的资源.

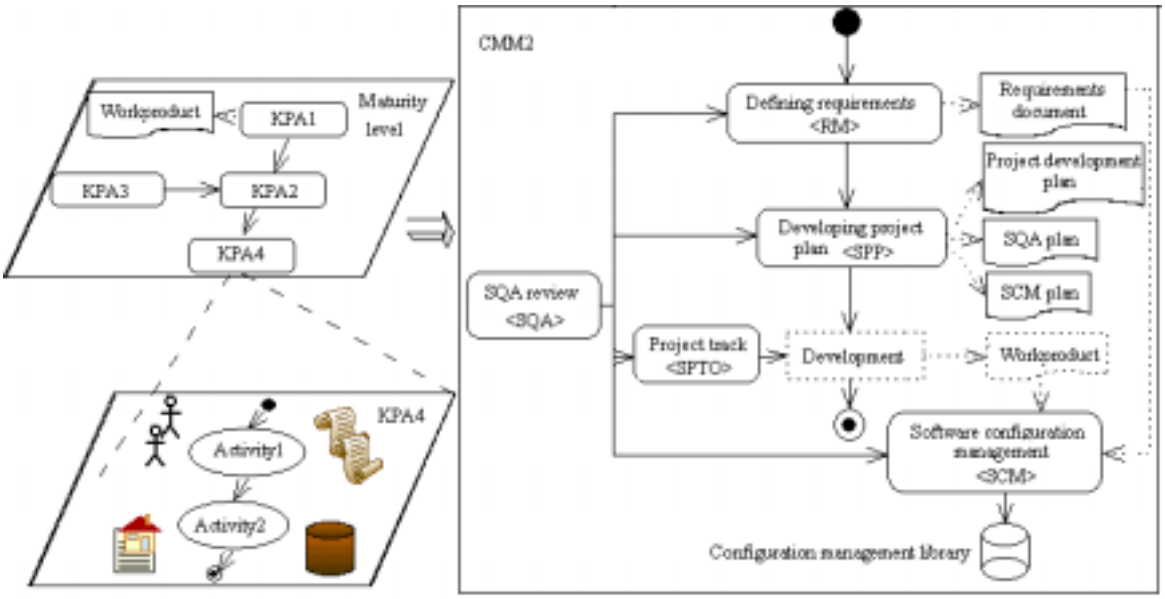

Fig.1 CMM software process

图 1 CMM 软件过程

1.2 CMM软件过程元模型的建模要求

元模型是描述模型构成的语言,而过程元模型是用于表示和组织软件过程模型的概念框架.MDA 框架的核心是 4 层元建模架构,如图 2 所示.在 4 层架构中上一层是用来描述下一层的语言,位于最高层 M3 层的是元对象设施 MOF(meta object facility)[15],又称为元元模型.MOF 是自描述的.基于 MOF 可以面向不同的领域定义不同的描述语言,即 M2 层的元模型,例如,软件过程工程元模型 SPEM 以及本文提出的 CMM 软件过程元模型 SPM-CMM.位于 M1 层的是模型层,例如,利用 SPEM 定义的过程模型 Rational 统一过程 RUP(rational unified process)和基于 SPM-CMM 定义的 CMM 软件过程模型就位于这一层.最底层 M0 层是包含了用户数据的、实际执行的过程.

本文的研究目标是在 MDA 框架下建立 CMM 软件过程元模型——SPM-CMM,这就要求 SPM-CMM 必须满足以下两个条件:

(1) SPM-CMM 应当符合 MOF.

MDA 框架为基于 MOF 的各种模型提供模型转换和数据交互的支持,其首要条件是所有的建模语言都应

当符合 MOF.

(2) SPM-CMM 应当支持 CMM 软件过程建模,既要体现 CMM 特定过程语义,也要具备表示可操作过程的能力.

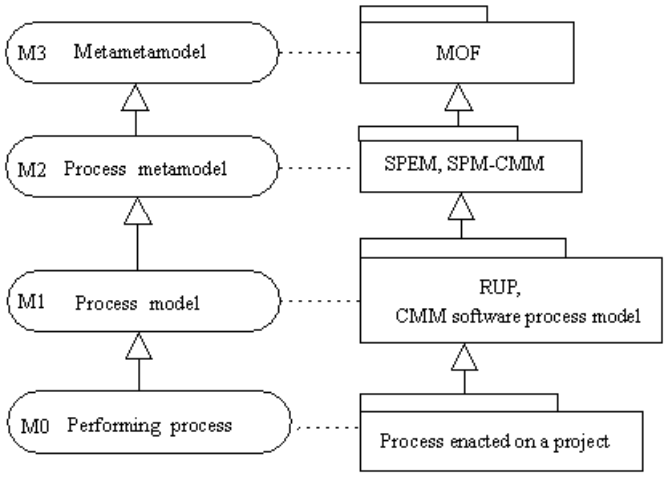

Fig.2 Four layers metamodeling architecture

图 2 4 层元建模架构

CMM 软件过程模型的作用在于以可操作的方式表达 CMM,以便企业能够结合自身特点实施过程改进活动.因此,作为 CMM 软件过程模型描述语言的 SPM-CMM,必须能够体现出 CMM 特定的过程语义,同时应当支持对一般软件过程进行建模.

1.3 面向CMM的SPEM扩展机制

由于不同的过程具有不同的特性,要解决过程模型一致的根本问题,必须建立统一的过程元模型.为此,OMG 提出了软件过程工程元模型 SPEM.SPEM 提取了 RUP 等多个软件开发过程中的公共特征,是一种软件过程的通用元模型.由于植根于 UML 概念,SPEM 易于为广大开发者和建模者所理解.同时,SPEM 支持 MDA 框架,提供模型和工具的互操作性.因此,本文选择 SPEM 作为 CMM 软件过程元模型的研究基础.

由于 SPEM 是通用的软件过程元模型,不能反映 CMM 特定的过程语义,为支持 CMM 建模,需要对 SPEM 进行扩展.重型扩展(heavyweight extension)和轻型扩展(lightweight extension)是 MDA 提供的两种扩展机制[16],轻型扩展得到的元模型可以获得 UML 工具的支持,但是扩展能力有限,不能在所定义的构造型之间以及已有的元类之间声明新的关联,无法利用 MOF 所提供的全部语义能力.而重型扩展具有强大的扩展能力,可以定义新的元类,但是无法获得 UML CASE 工具的支持.目前大多数扩展研究都基于单一扩展方式,例如,UML profile for J2EE 和 UML profile for .NET 都使用了轻型扩展机制,而重型扩展主要应用于软件体系架构建模[17]等方面.与以上研究不同的是,本文采用“重型+轻型”的扩展策略.重型扩展通过 MOF 扩展 SPEM,并定义新的元类以及元类之间的关联,最终建立 SPM-CMM.轻型扩展利用 Profile 机制进一步扩展 SPM-CMM,通过细化 SPM-CMM 的语义,对需要使用的子集作进一步的说明和约束,使 SPM-CMM 可以直接被 UML CASE 工具支持.这种策略的优点是既可以直接利用 SPEM 中丰富的过程元素,无须重新定义,也保证了与基于 SPEM 的工程过程模型进行数据交互时的转换效率.

2 一个 CMM 软件过程元模型 SPM-CMM

2.1 SPM-CMM

进行重型扩展的原则是:第一,不能修改 SPEM 原有的元素,不能改变其语法和语义;第二,尽可能地避免新定义的元类与已有的元类发生联系,从而维护 SPEM 原有语义和语法,保证 SPM-CMM 的独立性、完整性和可

操作性.

SPEM 元模型是扩展 UML 物理元模型的子集得到的,由 Foundation 和 Extension 两个包组成.Extension 包中定义的是软件过程建模所用的元素和语义.SPM-CMM 扩展了 Extension 中 3 个子包:ProcessStructure, ProcessLifecycle 和 ProcessComponent.SPM-CMM 对应 CmmElements 包,该包与其他包的关系如图 3 所示.

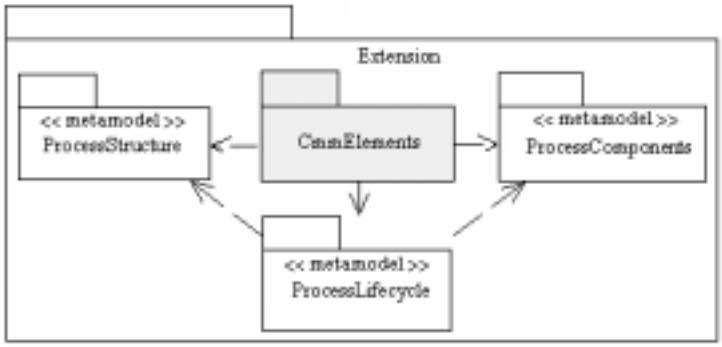

Fig.3 Relationships between the packages

图 3 包关系图

2.1.1 抽象语法

图 4 是 CmmElements 的抽象语法,包括 CmmElements 的元类及元类之间的关系,下面给出各个元类的说明.

(1) MaturityLevel

MaturityLevel 描述了软件过程能力的成熟度,具有 3 种属性:name,isLevel,achieved.其中 name 继承自元类 ModelElement.name 和 isLevel 是枚举类型,name 的取值范围是{Initial,Repeatable,Defined,Managed, Optimizing},IsLevel 的取值范围是{1,2,3,4,5}.achieved 是布尔型,表示该成熟度级别是否已达到.

(2) KPA

MaturityLevel 由多个 KPA 组成,KPA 描述的是一系列相关的活动,执行这些活动的目的是为了达到预定的目标,从而可以提高软件能力成熟度.具有属性 name 和 satisfied.对应于不同 MaturityLevel,name 有特定的取值范围.例如,当 MaturityLevel.isLevel=2 时,name 的取值范围为{RM,SPP,SPTO,SQA,SCM,SSM};satisfied 为布尔型,表示该 KPA 是否已满足.

(3) KpaGoal

KpaGoal 描述了 KPA 的范围和目的,实现 KPA 是为了达到一个或者多个 KpaGoal.具有属性 body(继承自元类 Constraint)和 achieved. body 用来描述 KpaGoal 的内容;属性 achieved 是布尔型,表示该目标是否达到.

(4) CommonFeature

CommonFeature 用来说明 KPA 实现的有效性和可重复性.具有属性 name.KPA 由 5 个 CommonFeature 组成,分别是 Commitment,MeasurmentAnalysis,VerifyingImplementation,ActivitiesPerformed 和 Ability.Commitment 是为建立过程,组织必须执行的活动;MeasurementAnalysis 是对过程进行的度量和分析活动; VerifyingImplementation 是验证执行过程与标准过程是否相符的活动;ActivitiesPerformed 描述了需要实现的活动;Ability 则是执行过程的前提条件,用来说明实现 KPA 的组织必须具备的能力,例如资源、技术、工具等,具有属性 body.

(5) KeyPractice

KeyPractice 描述了能够实现 KPA 的关键活动,具有属性 name.(4)中的 CommonFeature 均由多个 KeyPractice 组成.

(6) CmmAct

为了更好地表达 CMM 中特定的语义,使用 CmmAct 描述 CMM 专有的活动.具有属性 name,role,precondition,postcondition,input,output,step 和 resource.后 7 种属性是字符串型.

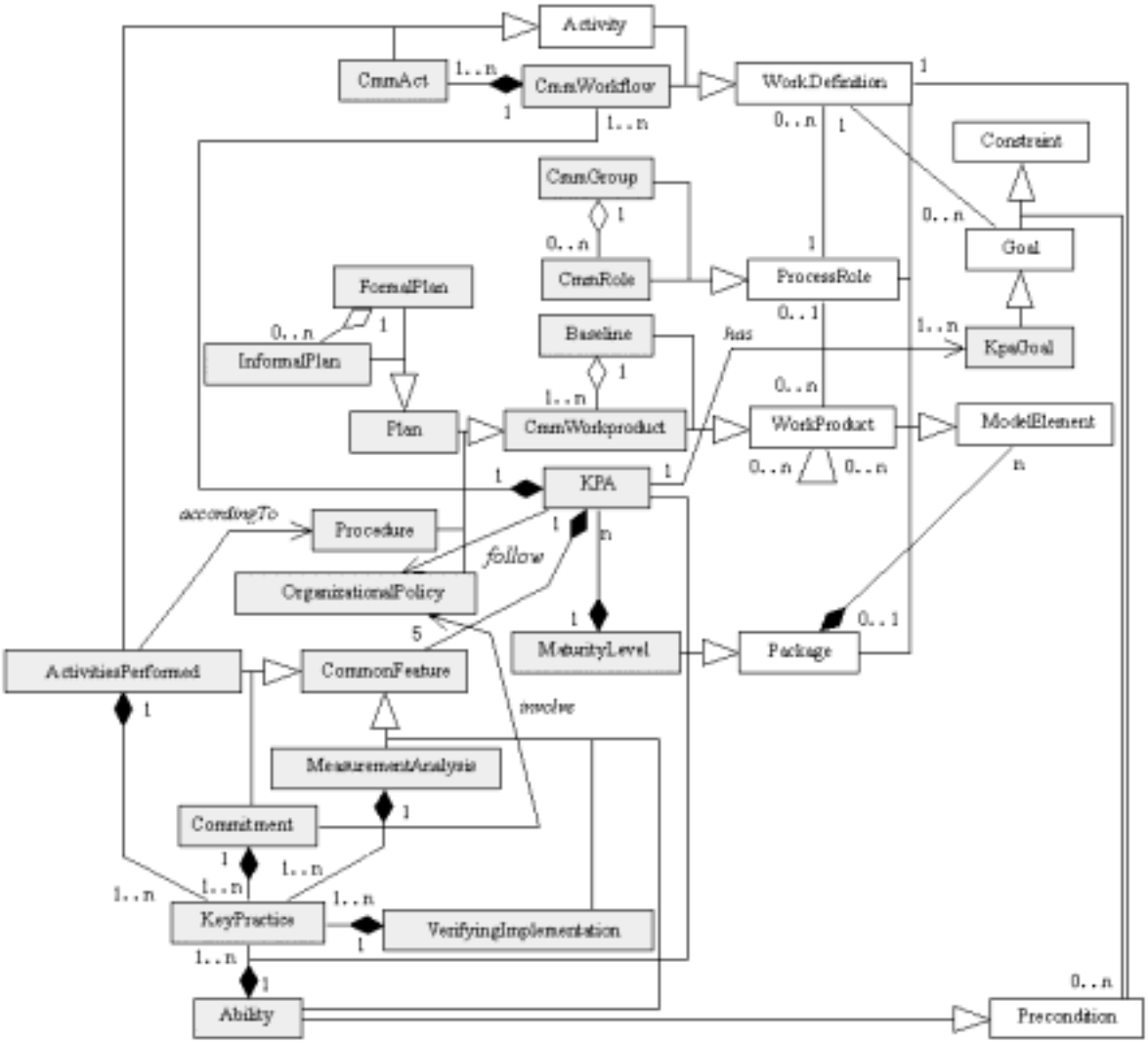

Fig.4 Abstract syntax of SPM-CMM

图 4 SPM-CMM 的抽象语法

(7) CmmProduct

CmmProduct 表示 CMM 中的工作产品,例如 RM 中的需求文档、SPP 中的软件开发计划等.具有属性 name 和 isDeliverable(继承自元类 WorkProduct).其中,计划 Plan 是一种重要的工作产品,分为 FormalPlan 和 InformalPlan.FormalPlan 表示的是必须制定的正式计划,例如项目开发计划、SQA 计划等.InformalPlan 则表示非正式的计划,例如风险管理计划、同行评审计划等.InformalPlan 可以作为 FormalPlan 的组成部分. 另外两种重要的 CmmProduct 是规程 Procedure 和组织方针 OrganizationalPolicy.

(8) Baseline

Baseline 表示 CMM 中基线的概念,基线是后续活动进行的基础,通常在软件过程里程碑处建立.具有属性 name 和 isDeliverable.

(9) CmmRole 和 CmmGroup

CmmRole 用于描述 CMM 中固定的角色,如项目经理等.CmmGroup 表示 CMM 中的组,如变更控制委员会 SCCB(software configuration control board)、软件工程组 SEG(software engineering group)等.CmmGroup 可以由多个 CmmRole 组成,例如 SCCB 就可以由项目经理和高级经理等角色组成.具有属性 name 和 responsibility(为

字符串型).

(10) CmmWorkflow

为了更好地划分 CMM 中的活动粒度,定义了 CmmWorkflow 来表示一系列活动的有序集.如在 RM 中,存在两个主要的 CmmWorkflow,分别是定义需求和管理需求.CmmWorkflow 可以由 Activity 及其关系组成.具有属性 name.

2.1.2 规则语义

SPM-CMM 的规则语义采用对象约束语言 OCL(object constraint language)描述.OCL 是用来表示约束的形式化语言,它的出现弥补了传统的形式化语言难以理解和使用的缺陷.OCL 定义了进行建模时必须满足的不变式条件.由于篇幅所限,本文只给出 SPM-CMM 的部分规则语义,针对部分元类,给出相应的不变式条件如下:

MaturityLevel:

[C1] To achieve a maturity level, the KPA for that level must be satisfied.

context *MaturityLevel* **inv**:

self.KPA→*forall*(*k*|*k.satisfied*=true) **implies** *self.achieved*=true

[C2] The name of level 1 is “Initial”. The name of level 2 is “Repeatable”. The name of Level 3 is “Defined”. The name of Level 4 is “Managed”. The name of Level 5 is “Optimizing”.

context *MaturityLevel* ***inv***:
if *isLevel*=1 **then**
name=Initial
endif
if *isLevel*=2 **then**
name=Repeatable
endif
if *isLevel*=3 **then**
name=Defined
endif
if *isLevel*=4 **then**
name=Managed
endif
if *isLevel*=5 **then**
name=Optimizing
endif

KPA:

[C3]Every KPA has 5 common features.

context *KPA* ***inv***:
self.commonFeature→*size*()=5

[C4] Every KPA is organized by more than 1 key practice.

context *KPA* ***inv***:
self.keyPractice→*size*()>1

[C5]All the KpaGoal must be achieved to satisfy that KPA.

context *KPA* ***inv***:
self.KpaGoal→*forall*(*g*|*g.achieved*=true) **implies** *self.satisfied*=true

[C6]A KPA is defined to reside at a single MaturityLevel.

context *KPA* ***inv***:
self.maturityLevle→*size*()=1

KeyPractice:

[C7]KeyPractice must follow an organizational policy.

context *KeyPractice* ***inv***:
self.organizationalPolicy.oclIsKindof(*OrganizationalPolicy*)
and *self.organizationalPolicy*→*size*()=1

2.2 SPM-CMM的UML Profile机制

通过 SPM-CMM 的 UML Profile 机制,可以使用 UML CASE 工具来建立和操作 CMM 软件过程模型.在本文中,UML Profile 说明了如何用 UML 元素来表示 SPM-CMM.一个 Profile 通常包括元素的构造型(stereotype)、标记值(tagged value)和约束(constraint).由于构造型提供了一种为元素分类的方式,因此,由这些构造型组成的

新元素,被称作虚元模型(virtual metamodel).SPM-CMM 的 UML Profile 虚元模型如图 5 所示,说明了构造型和标记值.在 Profile 中定义了构造型的图元,用来表示 CMM 软件过程模型.Profile 的约束利用 OCL 表示.由于篇幅所限,图元与约束在此不再进行详述.

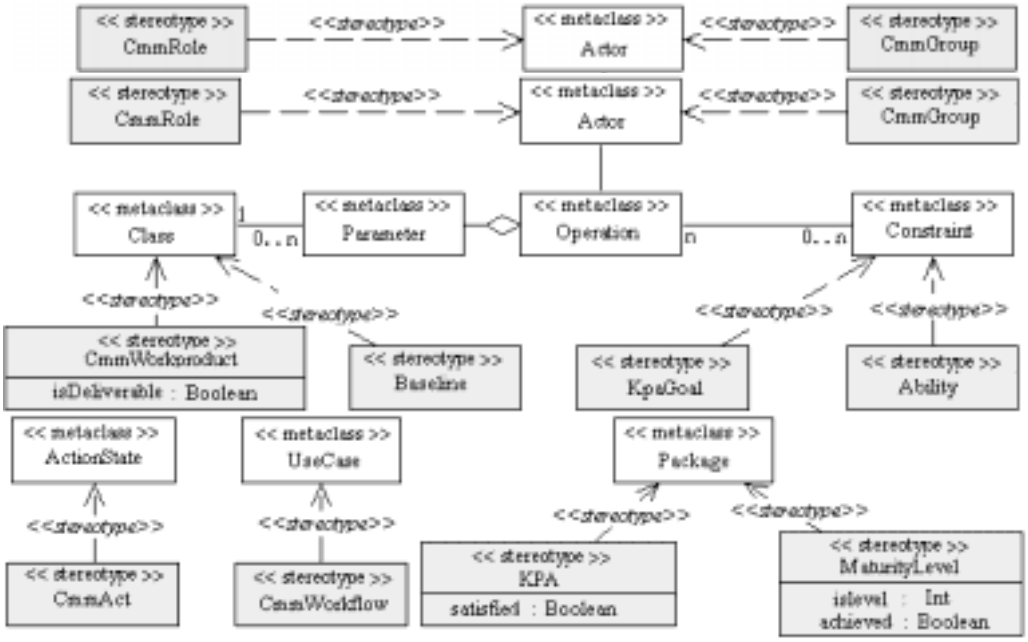

Fig.5 Virtual metamodel of SPM-CMM

图 5 SPM-CMM 的虚元模型

3 实例及工具支持

通过应用 SPM-CM 元模型,我们开发了支持 CMM 软件过程建模的原型工具 MDA-SPMT,如图 6 所示.

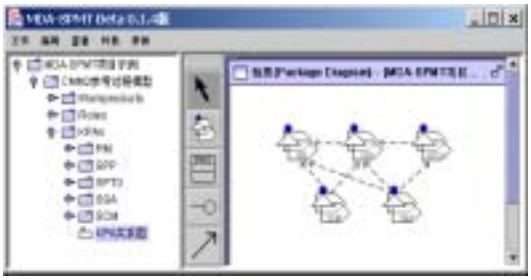

Fig.6 MDA-SPMT

图 6 MDA-SPMT

3.1 CMM软件过程模型

CMM 软件过程模型是基于 SPM-CMM 建立的,满足 SPM-CMM 的语法和规则语义.一方面,CMM 软件过程模型不反映具体企业的组织特征和过程特点,是一个指导 CMM 实施的通用过程模型,使用图形化的方式来表示该过程模型,有利于用户深入理解 CMM 体系.具体内容可参见下文的实例.另一方面,CMM 软件过程模型是最终获取特定企业的 CMM 实施过程模型的基础,后者是 CMM 软件过程模型与企业特征模型相结合的产物.例如,在关键过程域需求管理中,软件过程模型中所定义的重要工作产品是“需求文档”,而对于一个特定的企业,“需求文档”可映射为该企业需求过程中的“用例规约”、“补充规约”以及“界面原型”,而与“需求文档”相关的

关键实践,如“评审”等,也就相应地构成了企业 CMM 实施过程模型的重要组成部分.

3.2 一个实例

CMM 软件过程模型按照角色、工作产品以及 KPA 进行组织,角色是已定义职责的单元,可以分配给一个或多个个体,工作产品是活动成果的表现形式,角色和工作产品由 KPA 中的活动进行关联.以 CMM2 可定义级为例,选取 5 个 KPA 进行建模,分别是:需求管理 RM、配置管理 SCM、软件质量保证 SQA、项目策划 SPP 和项目计划跟踪与评估 SPTO.如图 6 左部所示,CMM2 参考过程模型分为 3 个包:Roles 角色包、Workproducts 工作产品包和 KPA 关键过程域包.图 6 右部所示为 5 个 KPA 之间的关系图,描述了 KPA 之间的依赖关系,例如 SPTO 与 SPP 相关,因为跟踪活动必须依据软件开发计划进行, SPP 与 RM 相关,因为软件开发计划必须基于确定的需求制定.

3.2.1 Roles

角色包定义的是 CMM 中所使用的角色.这些角色承担了特定的职责和任务,例如项目经理负责控制、监督项目的进行,并最终对客户负责,软件质量保证人员进行过程评审和工作产品审计,支持协助项目经理的部分管理活动.

角色包细分为 3 个子包:Manager,Groups 和 OtherRoles.在 Manager 中定义的是经理类角色,例如高级经理、项目经理、项目软件经理、一线软件经理等,其职责任务在角色属性中进行描述.在 Groups 中定义的是组类角色,例如质量保证组、软件配置控制委员会、软件配置管理组等,这些组由某些部门、经理和个人组成,共同完成一定的任务.在 OtherRoles 中定义的是其他类的角色,例如项目成员、任务负责人等.除了角色,在这些包中还定义了与角色相关的工作产品.图 7 是各子包中部分角色及其工作产品图,例如,项目经理负责制定软件开发计划,编写项目报告等.为增强 CMM 软件过程模型的实用性,进一步明确与 CMM 中重要工作产品相关的负责人和使用者,在 CMM 所定义角色的基础上增加了部分角色,例如“需求工程师”、“变更申请人”等,需求工程师负责编写需求文档,变更申请人填写基线变更申请.

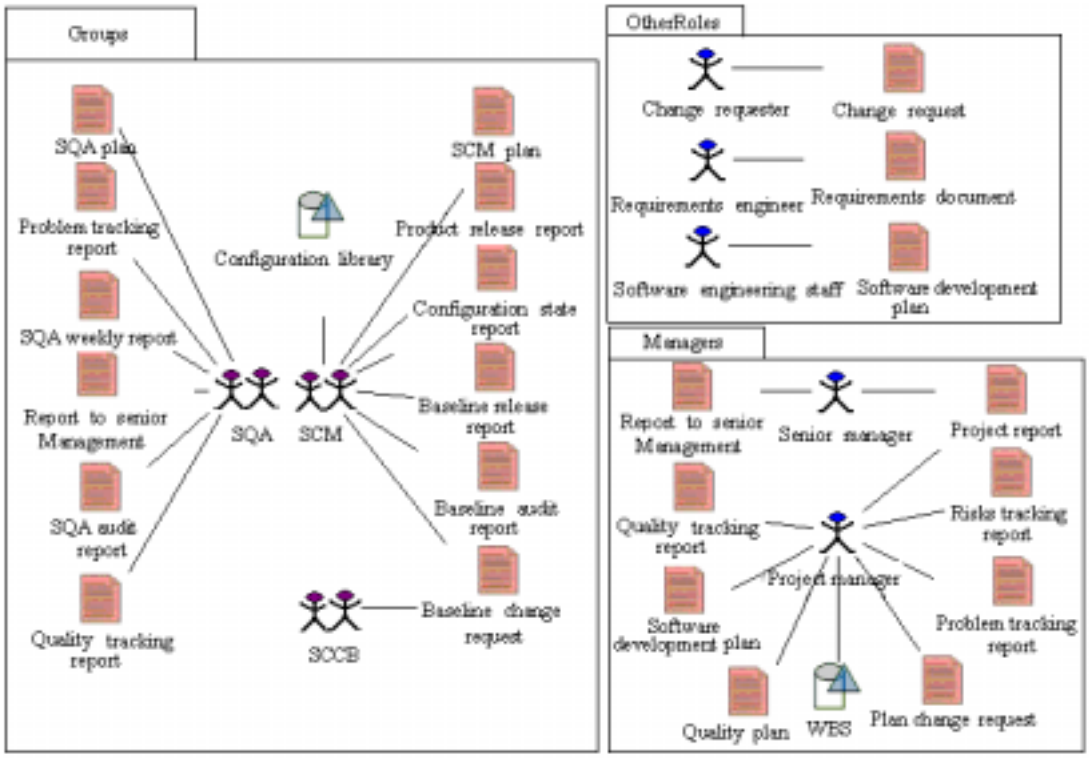

Fig.7 Roles package of CMM2 software process model

图 7 CMM2 软件过程模型的 Roles 包

3.2.2 Workproducts

CMM 的 KPA 定义了执行关键活动所生成或者使用的工作产品,在 CMM 软件过程模型中进一步明确和细化这些工作产品.实现 CMM 所需使用的产品形式定义包含于工作产品包中.工作产品包按照 KPA 进行了划分,如图 8 所示,CMM2 软件过程模型的工作产品包共分为 5 个子包,每个子包里定义的是实现该 KPA 中所需要的工作产品,例如,在 SQA 包中定义了软件质量保证计划、过程评审报告、产品审计报告、质量跟踪报告等工作产品.软件质量保证人员在项目确定初期策划软件质量保证活动,并制定软件质量保证计划.在项目进行的过程中,执行过程评审和产品审计后,形成相应的评审、审计报告,对发现的问题进行跟踪,形成质量跟踪报告.针对每个工作产品,在其属性中说明形成基线、评审等使用建议,例如在需求管理中,对所形成的需求文档,应当由相关人员进行评审,并形成需求基线,纳入软件配置管理.

Fig.8 Workproducts package of CMM2 software process model

图 8 CMM2 软件过程模型的 Workproducts 包

3.2.3 KPA

KPA 是对提高软件能力成熟度起到关键作用的软件过程.CMM 软件过程模型在 KPA 包中规定了各 KPA 之间的依赖关系,并针对每个 KPA,定义了其核心工作流.核心工作流满足 KPA 的所有目标要求,目标要求在属性中进行说明.与各核心工作流相关的角色也在包中进行定义.图 9 所示左部显示的是配置管理 SCM 所包含的 7 个核心工作流,分别为制定配置管理计划、维护配置库、记录配置状态、配置变更控制、基线发布控制、产品发布控制和配置审计.具体执行和参与这些核心工作流的角色为配置管理员、项目经理、软件配置控制委员会以及质量保证组.针对每个核心工作流,CMM 软件过程模型定义了具体的活动执行流程,整个流程由活动元素组成,在活动属性中说明执行该活动的角色职责、入口\出口准则、输入\输出工作产品、步骤、资源能力要求以及度量元等信息.图 9 右部是进行配置变更控制时建议执行的活动流程,例如,提交变更申请后,必须由软件配置控制委员会评估变更申请,批准后才能进行基线变更.具体变更的情况应当进行跟踪验证,最后由配置管理员向各相关组和人员发布变更通知.

3.2.4 规则约束

如前文所述,我们采用 OCL 语言来描述 CMM 软件过程模型的规则约束.规则约束规定了活动执行必须满足的不变式条件,例如“基线变更申请必须由软件配置控制委员会进行评审”、“软件配置管理人员应接受配置管理相关培训”等.规则约束保证 CMM 软件过程模型的精确性,并为后期的模型转换提供语义支持.以配置管理为例,下面是其部分规则约束.

BaselineChangesRequest:

[R1] Changes to baseline are reviewed by SCCB.

context *BaselineChangesRequest* ***inv***:

self.reviewer.oclIsKindof(*SCCB*)

SCM:

[R2] SCM are trained to perform SCM activities.

context *SCM* ***inv***: *self.trained*=true

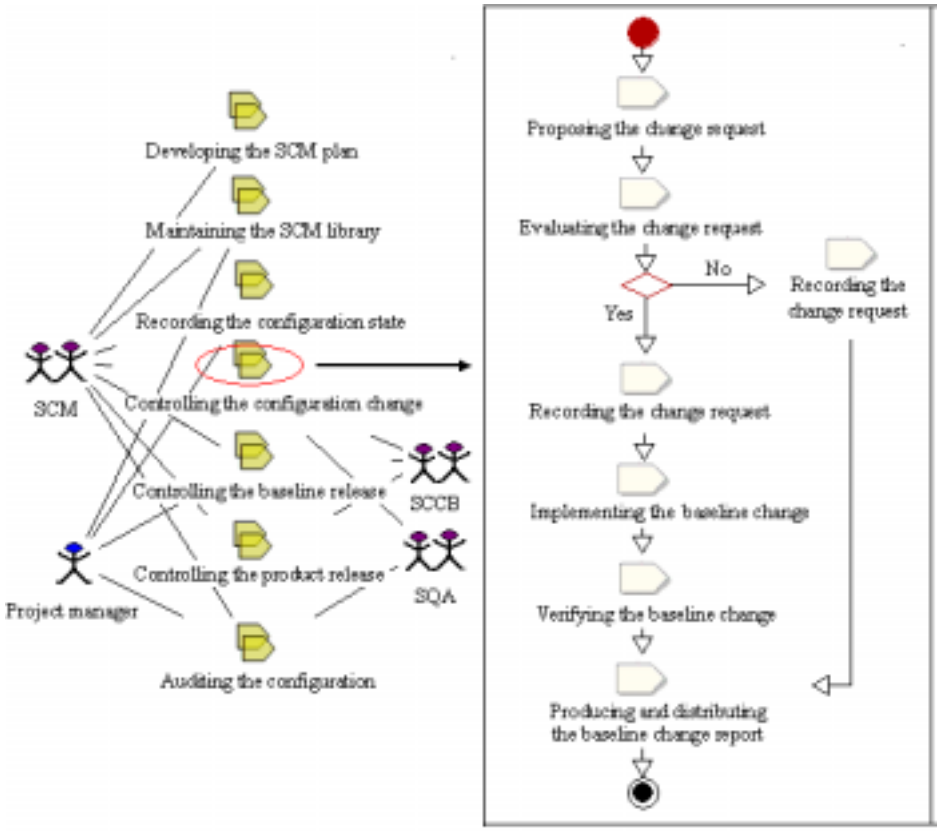

Fig.9 SCM of CMM2 software process model

图 9 CMM2 软件过程模型的 SCM

4 总 结

CMM 的广泛应用证明了其在软件过程改进中起到的巨大作用,然而企业要实施 CMM 仍然面临诸多困难.其中一个主要障碍在于如何应用 CMM 评估体系,使之成为能够反映企业组织结构和过程特点的 CMM 实施过程模型.我们认为,利用模型驱动架构 MDA 可以解决 CMM 软件过程模型向 CMM 实施过程模型过程转换的问题.为此,本文提出了一个 CMM 软件过程元模型——SPM-CMM.该元模型扩展了软件过程工程元模型 SPEM,有效地利用了 SPEM 中丰富的过程语义,同时也支持 CMM 特定过程语义.在此基础上,我们实现了一个用于 CMM 软件过程建模的原型工具 MDA-SPMT.

在进一步的工作中,我们将就过程模型转换问题进行研究,即 CMM 软件过程模型如何自动或半自动地向企业 CMM 实施过程模型进行转换,并给出模型转换和模型验证的方法.2002 年,SEI 综合不同领域的 CMM,提出了 CMMI(capability maturity model integration)模型[18].由于 CMMI 模型以 CMM 为基础,并具备过程特点,SPM-CMM 元模型经过进一步扩展也可以应用于 CMMI 过程建模,这也是有待研究的问题.

致谢 在此,我们向对本文的工作给予支持和建议的王永吉研究员和实验室导师小组的全体老师,以及袁峰、赵欣培、徐伟同学表示感谢.

References:

[1] Herbsleb JD, Goldenson DR. A systematic survey of CMM experience and results. In: Rombach HD, ed. Proc. of the 18th Int'l Conf. on Software Engineering (ICSE'96). Washington: IEEE Computer Society, 1996. 323−330.

[2] Jalote P. Lessons learned in framework-based software process improvement. In: Gupta G, Reed K, eds. Proc. of the 9th Asia-Pacific Software Engineering Conference (APSEC 2002). Washington: IEEE Computer Society, 2002. 261−265.

[3] Hall T, Rainer A, Baddoo N. Implementing software process improvement: An empirical study. Software Process: Improvement and Practice, 2002,7(1):3−15.

[4] Ginsberg MP, Quinn LH. Process tailoring and the software capability maturity model. CMU/SEI-94-TR-024: Pittsburgh: Software Engineering Institute, Carnegie Mellon University, 1995. 1−60.

[5] Straub P, Guzmán D. Incremental, collaborative software process improvement in a tiny software group. In: Campos AE, ed. Proc. of the XXII Int'l Conf. of the Chilean Computer Science Society (SCCC 2002). Washington: IEEE Computer Society, 2002. 187−194.

[6] Guerrero F, Eterovic Y. Adopting the SW-CMM in a small IT organization. IEEE Software, 2004,21(4):29−35.

[7] Keenan F. Agile process tailoring and problem analysis (APTLY). In: Gould C, Su ZD, Premkumar D, eds. Proc. of the 26th Int'l Conf. on Software Engineering (ICSE 2004). Washington: IEEE Computer Society, 2004. 45−47.

[8] Miller J, Mukerji J, eds. Model driven architecture. ormsc/2001-07-01: Needham, Object Management Group, 2001. 1−31. http://www.omg.org/cgi-bin/doc?ormsc/2001-07-01

[9] Zhao XP, Li MS, Wang Q, Chan K, Leung H. An Agent-based self-adaptive software process model. Journal of Software, 2004, 15(3):348−359 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/348.htm

[10] Jäger D, Schleicher A, Westfechtel B. Using UML for software process modeling. ACM SIGSOFT Software Engineering Notes, 1999,24(6):91−108.

[11] Alexandre S, Habra N. UML modeling of five process improvement models. LQL-2003-TR-02: Charleroi, CETIC-FUNDP, Namur, 2003. 1−42. http://www.cetic.be/IMG/pdf/UMLModelingOfFiveProcessModels-V1-3.pdf

[12] Object Management Group. Software process engineering metamodel specification. Version 1.0, formal/02-11-14: Needham, Object Management Group, 2002. 1−98. http://www.omg.org/cgi-bin/doc?formal/2002-11-14

[13] Breton E, Bézivin J. Process-Centered model engineering. In: Wang GJ, Lupu CE, Wegmann A, eds. Proc. of the 15th IEEE Int'l Enterprise Distributed Object Computing Conf. Washington: IEEE Computer Society, 2001. 179−182.

[14] Mark CP, Bill C, Mary BC, Charles VW. Capability maturity model for software. Version 1.1. CMU/SEI-93-TR-024: Pittsburgh, Software Engineering Institute, Carnegie Mellon University, 1993. 1−82. http://www.sei.cmu.edu/publications/documents/93.reports/93.tr.024.html

[15] Object Management Group. Meta object facility (MOF) specification. Version1.4, formal/02-04-03: Needham, Object Management Group, 2002. http://doc.omg.org/formal/02-04-03

[16] Frankel DS; Bao ZY, Trans. Model Driven Architecture: Applying MDA to Enterprise Computing. Beijing: Posts & Telecom Press, 2003. 145−161 (in Chinese).

[17] Pérez-Martínez JE. Heavyweight extensions to the UML metamodel to describe the C3 architectural style. ACM SIGSOFT Software Engineering Notes, 2003,28(3):1−5.

[18] CMMI Product Team. CMMI for Systems Engineering/Software Engineering/Integrated Product and Process Development/Supplier Sourcing, Version1.1, Continuous Representation (CMU/SEI-2002-TR-011) and Staged Representation (CMU/SEI-2002-TR-012): Pittsburgh, Software Engineering Institute, Carnegie Mellon University, 2002. http://www.sei.cmu.edu/cmmi/models/models.html

附中文参考文献:

[9] 赵欣培,李明树,王青,陈振冲,梁金能.一种基于 Agent 的自适应软件过程模型.软件学报,2004,15(3):348−359. http://www.jos.org.cn/1000-9825/15/348.htm

[16] Frankel DS,著;鲍志云,译.应用 MDA.北京:人民邮电出版社,2003.145−161.