一种基于模型融合的 CMM 实施过程建模方法

李 娟[1),2)] 袁 峰[1),2)] 李明树[1),3)] 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 100039)

[3)](中国科学院软件研究所计算机科学重点实验室 北京 100080)

摘 要 提出了一种基于模型融合的 CMM 实施过程建模方法. 该方法使用软件过程工程元模型 SPEM 建立 CMM 过程模型 CPM 和企业过程模型 EPM,通过融合 CPM 和 EPM 来获得 CMM 实施过程模型 CIPM. 文中利用带标记的有向图描述过程模型,给出了模型融合方法,并进行了一致性证明. 最后通过一个过程模型融合原型工具和实例说明了方法的应用情况.

关键词 CMM;软件过程建模;模型融合;有向图

中图法分类号 TP311

A Model Merging Based Approach for Modeling the CMM Implementation Process

LI Juan[1),2)] YUAN Feng[1),2)] LI Ming-Shu[1),3)] WANG Qing[1)]

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

[2)](*Graduate School of the Chinese Academy of Sciences, Beijing* 100039)

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

Abstract As a widely used software process improvement model, CMM (Capability Maturity Model) provides a guide for choosing process improvement strategies by facilitating the identification of current process capabilities. However, how to model the CMM implementation process remains an open issue for a fact that enterprise software processes are complex and variable. This paper presents an approach for modeling the CMM implementation process based on model merging mechanism. Using SPEM (Software Process Engineering Metamodel), CPM (CMM Process Model) and EPM (Enterprise Process Model) can be built and then merged to achieve CIPM (CMM Implementation Process Model). Also when EPM is changed, the changed content can be reflected conveniently in CIPM using the model merging mechanism. In above model merging approach, process models are described with the labeled directed graph and merging algorithms are proposed. Authors prove model merging consistency and use a process model merging prototype tool to show that the proposed approach is practical and effective.

Keywords CMM; software process modeling; model merging; directed graph

---

收稿日期:2005-01-31;**修改稿收到日期**:2005-10-24. **本课题得到国家自然科学基金**(60273026,60473060)、**国家"八六三"高技术研究发展计划项目基金**(2002AA116060,2001AA113080)**资助**.**李 娟**,女,1977 年生,博士研究生,主要研究方向为软件过程改进、模型驱动架构、需求工程. E-mail: lijuan@itechs.iscas.ac.cn. **袁 峰**,1977 年生,博士研究生,主要研究方向为软件过程建模、模型驱动架构. **李明树**,1966 年生,研究员,博士生导师,主要研究领域为智能软件工程、实时系统. **王 青**,女,1964 年生,研究员,博士生导师,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.

1 引　言

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 的关键作用在于提供了有效的模型来辅助企业评估和改进软件过程. 一个企业要进行 CMM 过程改进，就必须基于 CMM 体系，结合企业过程特征建立 CMM 实施过程. 目前应用 CMM 主要依靠评估师对目标企业进行评估，考察分析企业的过程特点，从而提出相应的过程改进方案，整个过程需要大量的人力、物力支持. 由于企业外部环境的日益复杂以及市场需求的频繁变化，当企业自身的组织结构和业务过程发生改变时，又会造成 CMM 实施过程的频繁变动，需要进行反复的 CMM 自评估来保证其正确性，因而导致 CMM 过程改进成本的增加. CMM 实施过程建模由于其复杂性和易变性，已经成为软件过程改进中的难点问题.

模型融合(model merging)是解决复杂建模问题的有效方法[1]. 这种方法通过把从不同的角度对问题域建立的模型进行融合来得到最终的模型，避免了直接建模的复杂性. CMM 提供了通用的过程改进指导，而企业过程模型中则包含了企业过程特征，将二者进行模型融合，可以得到符合 CMM 的、反映企业过程特征的 CMM 实施过程模型. 并且，当企业过程发生变化的时候，基于融合机制可以将这种变化及时反映到 CMM 实施过程模型中，保证 CMM 实施过程模型的正确性，提高实施效率和质量，节省成本，从而有效支持企业 CMM 过程改进以及进行 CMM 自评估.

模型融合主要分为两类：基于相同元模型的模型间融合和基于不同元模型的模型间融合. 本文将前者称为“同元”模型融合，后者称为“异元”模型融合. “异元”模型融合主要通过定义元模型之间的映射关系来保证一致性，如数据模型融合及冲突解决方法[2,3]、本体模型融合中基于匹配的融合方法[4]以及视图模型融合中的多视图统一的方法[5]等. “异元”模型融合由于要处理元模型之间的不一致，因而在应用中较为繁琐和困难. 而“同元”模型融合中，待融合模型都基于相同的元模型，可以保证融合中各模型元类(meta class)的一致性，减少语义冲突，具有更好的可操作性[6,7].

本文提出一种基于模型融合的 CMM 实施过程建模方法，如图 1 所示. 该方法属于“同元”模型融合方法，同时也借鉴了“异元”模型融合中的相关技术. 首先，选择软件过程工程元模型 SPEM(Software Process Engineering Metamodel)[8]作为建模基础，分别建立 CMM 过程模型 CPM(CMM Process Model)和企业过程模型 EPM(Enterprise Process Model)，其中 CPM 依据 CMM 体系定义，描述 CMM 中提供的通用的软件过程改进模型，EPM 描述的是特定企业的过程模型，反映企业过程现状；然后，通过模型融合得到 CMM 实施过程模型 CIPM(CMM Implementation Process Model). CIPM 既满足 CMM 对过程改进的约束，也包含企业特有的过程特征. 该方法用带标记的有向图表示过程模型，利用活动片段(activity segment)对过程模型进一步抽象，以 CPM 为基础，使用多种融合算法确定 EPM 与 CPM 中的不一致情况并进行调整，最终得到 CIPM.

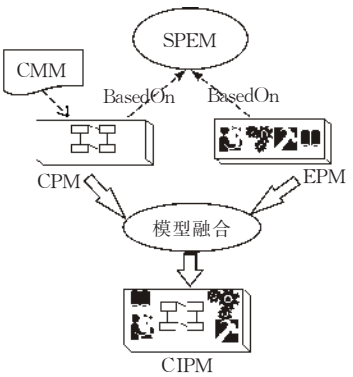


图 1　CPM 和 EPM 的模型融合

文中第 2 节分析融合方法中的过程模型，包括作为建模基础的 SPEM 元模型以及 CPM、EPM 和 CIPM 过程模型；第 3 节在给出过程模型融合相关概念的基础上，说明模型融合过程中的一致性问题，并给出具体的过程模型融合方法及模型融合一致性证明；第 4 节通过一个原型工具和实例说明融合方法的应用情况；最后，第 5 节总结全文并指出进一步的研究方向.

2 融合方法中过程模型分析

本文的融合方法共涉及 3 种过程模型：CPM、EPM 和 CIPM. 这 3 种模型均基于 SPEM 定义，采用了相同的表示形式，但其模型内容各有不同. CPM 依据 CMM 体系建立，反映 CMM 对过程改进

的一般性约束，是企业进行过程改进的依据；EPM 针对特定企业进行过程建模，表示企业当前过程特征，是企业进行过程改进的对象；CIPM 是通过融合 CPM 和 EPM 得到的，既体现企业现有的过程特征，又满足 CMM 一般性约束，是企业进行过程改进的结果. 下面首先介绍用来描述 CPM、EPM 和 CIPM 的软件过程工程元模型 SPEM.

2.1 软件过程工程元模型 SPEM

软件过程工程元模型 SPEM 是对象管理组织 OMG(Object Management Group)提出的，其目的是统一具有不同特性的软件过程，解决过程模型不一致的问题. SPEM 提取了 Rational 统一过程 RUP (Rational Unified Process)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. 使用 SPEM 表示软件过程，如 RUP 和 XP(eXtreme Programming)[9]，有利于过程的抽象表示以及过程复用. 因此，我们选择 SPEM 作为 CPM、EPM 和 CIPM 的建模基础.

SPEM 是一种以活动为中心的建模语言，其核心概念模型是角色、工作产品和活动. 由于 SPEM 本身较为繁琐，为方便论述，本文仅说明其核心概念模型，并明确本文所使用的语义约束. 如图 2 所示，SPEM 概念模型中包括 3 种元素：角色(Role)、活动(Activity)和工作产品(WorkProduct). 角色执行活动，工作产品输入活动，活动输出工作产品. 其中，3 种元素均具有 Name 属性，表示元素的名称，工作产品还具有 State 属性，表示经过活动处理之后工作产品的状态. 通过定义状态可以更清晰地说明同一工作产品在整个生命周期中的变化. State 的值可为空. 一个工作产品可以有多个状态，多个工作产品的状态可以相同.

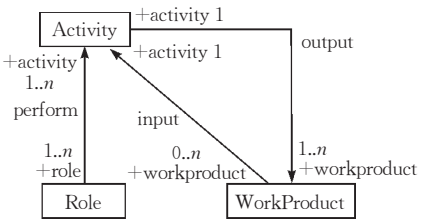


图 2 SPEM 的核心概念模型

各模型元素的约束关系主要有：(1)所有元素的名称均唯一；(2)一个具有特定状态的工作产品只能由一个活动输出，一个活动至少输出一个具有特定状态的工作产品；(3)一个活动至少由一个角色执行，一个角色至少执行一个活动.

2.2 CPM、EPM 和 CIPM

2.2.1 CMM 过程模型 CPM

依据 CMM 体系，CPM 中规定了一系列的 CMM 过程元素以及元素间关系，提供了过程改进的参考模型. CPM 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，以易于理解的、过程化的方式进行表示.

CPM 可以从“成熟度级别”和“关键过程域”两个层次来说明. 从“成熟度级别”这一层来看，CPM 中表示了各关键过程域间的依赖关系. 以 CMM2 中的五个关键过程域为例，分别是需求管理 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). RM 中确定需求之后，开始进行 SPP 制定项目计划，SPP 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，应定期进行审计和评审，以保证整个项目按照计划执行. 从“关键过程域”这一层来看，CPM 中表示了各关键过程域所包含的一系列相关的活动、人员、目标、约束条件以及资源等. 另外，CPM 利用对象约束语言 OCL(Object Constraint Language)[10]定义了约束规则，例如裁剪规则. 在文献[11]中，我们基于 SPEM 给出了 CMM 过程模型，本文不再详述.

2.2.2 企业过程模型 EPM

EPM 体现企业当前过程特征，表示企业软件开发与管理的现状. EPM 反映了 CMM 评估所关注的企业软件过程元素及其关系，包括：(1)开发过程元素，例如项目计划、需求定义和设计等；(2)支持过程元素，例如配置管理、质量保证和组间协调等；(3)组织管理过程元素，例如高层管理、培训和基础设施等. EPM 所包含的过程元素比 CPM 丰富，因为 EPM 是企业完整的过程模型，而 CPM 仅与其中的某些关键过程相关. 按照 EPM 与 CPM 的关系，EPM 中的过程元素可分为两类，即 CMM 类型元素(CMM-typed Element)和自由元素(Free Element)，如表 1 所示. 模型融合需要检查和调整 EPM 中的 CMM 类型元素及其关系，使其满足 CPM 的约束.

表 1　EPM 元素类型

元素类型	定义	举例
CMM 类型元素	EPM 中可与 CPM 元素对应的过程元素，是 CPM 中的过程元素在企业实施过程中的具体表现形式.	EPM 中的“UseCase(用例)规约”是 CMM 类型元素，与 CPM 中的“需求规约”对应.
自由元素	EPM 中与 CPM 元素不对应的过程元素，表示企业特有的过程特征.	EPM 中的“测试文档”为自由元素，因为 CPM 中没有关于测试的过程元素定义.

2.2.3　CMM 实施过程模型 CIPM

CIPM 是 CPM 与 EPM 模型融合后得到的符合 CMM 的企业过程模型. 实质上，模型融合是依据 CPM，对 EPM 进行的调整操作，这种调整包括在 EPM 中增加缺少的 CMM 类型元素和关系、修改元素的关系、删除冲突的关系等. 由于 CIPM 融合了 CPM 和 EPM 两个过程模型的特点，既具备企业过程完整的过程特征，也在关键过程上满足 CMM 的约束，因此，CIPM 可以直接用于指导企业软件开发，并且当 EPM 发生改变时，利用融合机制可以及时调整 CIPM，保证 CMM 过程改进能够满足企业的业务需要，提高过程改进的效率. 另外，由于 CIPM 采用 SPEM 表示，易于裁剪、维护、发布、实施和复用.

CIPM 应当满足以下一致性约束：

(1)所有过程元素及其关系均满足 SPEM 元模型约束. CIPM 中所有元素均只应当是 Role、Activity 和 WorkProduct 等元类的实例，CIPM 中所有元素的关系均满足 SPEM 中关于元素关系的定义.

(2)保留 CPM 和 EPM 中必要的过程元素. CIPM 中应当保留以下过程元素：CMM 中所有不可裁剪的过程元素、EPM 中所有过程元素. 对于 CMM 中的可裁剪元素，如果该元素在 EPM 没有匹配项，则不应出现在 CIPM 中.

(3) CIPM 中的 CMM 类型过程元素关系与 CPM 一致. 如前所述，EPM 与 CPM 之间存在绝对不一致和相对不一致，模型融合就是为了消除这些不一致，因此所得到的 CIPM 中的 CMM 类型过程元素关系应与 CPM 保持一致.

(4)CIPM 中的自由元素间的关系与 EPM 一致. CIPM 中的自由元素来自 EPM，不受 CPM 约束，因此，CIPM 中的自由元素关系应当与 EPM 保持一致.

(5)CIPM 中不包含 CPM、EPM 以外的过程元素和关系. 所有 CIPM 中的元素和关系均应当是 CPM 和 EPM 中定义的.

3　CMM 实施过程建模的模型融合方法

3.1　基本定义

3.1.1　过程模型有向图

定义 1.　一个过程模型的有向图是一个六元组：$G=(V,E,L_V,L_E,\mu,\nu)$，其中：

V 为顶点的集合；

$E\subseteq V\times V$，为边的集合，顶点 V_i 到 V_j 的有向边表示为 $E(V_i,V_j)$；

L_V 是顶点的标记集合；

L_E 是边的标记集合；

μ 是从顶点集合 V 到标记集合 L_V 的映射，用于对每个顶点赋一个标记. 各顶点的标记是唯一的；

ν 是从边集合 E 到标记集合 L_E 的映射，用于对每个边赋一个标记；

顶点 V_1 到 V_n 之间的路径是一个集合，表示为 $R(V_1,V_n)=\{(V_1,V_i,\cdots,V_m,V_n),\cdots,(V_1,V_x,\cdots,V_w,V_n)\}$. 其中 $(V_1,V_i,\cdots,V_m,V_n)$ 等表示具体路径，用经过的所有顶点表示，并且 V_1 到 V_n 的任意两条路径，中间经过的顶点至少有一个是不同的.

基于 SPEM 的过程模型可用带标记的有向图描述. 有向图中边的标记有 3 种：HasV，NoV 和 Exist. HasV 表示两个顶点之间可出现其它顶点，NoV 表示两个顶点之间不出现其它顶点，Exist 表示两个顶点之间可出现有向边.

3.1.2　活动片段

基于 SPEM 的过程模型由于过程元素的类型及关系繁杂，因而融合操作复杂. 为了对过程模型进行简化，我们引入活动片段的概念，通过将一个活动及其相关角色和工作产品共同抽象为有向图中的一个顶点，从而简化过程模型有向图，提高模型融合的可操作性.

定义 2.　活动片段是一个三元组 $AS=(A,R,W)$. 其中：

A 为一个活动；

R 为与 A 相关的所有角色的集合；

W 为 A 输出的所有工作产品及其状态的集合. 工作产品 W_x 的状态集合记为 $State(W_x)$. 状态为 s 的工作产品 W_x 表示为 $W_x.Name\langle s\rangle$,简记为 $W_x\langle s\rangle$,其中 $s\in State(W_x)$.

如果活动片段 AS 中的活动为 CMM 类型元素,则 AS 称为 CMM 类型片段,如果为自由元素,则 AS 称为自由片段. 采用活动片段简化后的过程模型有向图中,顶点代表活动片段,边代表活动片段之间的有向关联. 如图 3 所示,图上方的过程模型中有 3 个活动片段:$AS_1=\{(A1),(R1),(W1\langle S1\rangle)\}$,$AS_2=\{(A2),(R1),(W2\langle S1\rangle)\}$,$AS_3=\{(A3),(R2),(W1\langle S2\rangle,W2\langle S2\rangle)\}$. 图下方为过程模型有向图.

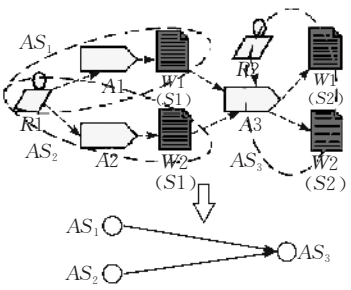


图 3 活动片段

3.1.3 融合中的匹配方式

模型融合使用元素匹配和活动片段匹配两种方式. 元素匹配又分为完全匹配和半匹配两种情况.

定义 3. 当且仅当满足下列匹配规则时,称 CPM 中的元素 e 与 EPM 中的元素 e' 完全匹配,记为 $e\leftrightarrow e'$. $Metaclass(e)$ 表示 e 的元类,$\approx$ 表示语义相同或者相近,反之记为 $\not\approx$. 有以下匹配规则:

(1)$(Metaclass(e)=(\text{'Role'}\vee\text{'Activity'}))\wedge(Metaclass(e)=Metaclass(e'))\wedge(e.Name\approx e'.Name)\Rightarrow e\leftrightarrow e'$;

(2)$(Metaclass(e)=\text{'WorkProduct'})\wedge(Metaclass(e)=Metaclass(e'))\wedge(e.Name\approx e'.Name)\wedge(\forall s(\exists s's'\approx s)(s\in State(e)s'\in State(e')))\Rightarrow e\leftrightarrow e'$.

定义 4. 当且仅当满足下列匹配规则时,称 EPM 中的元素 e' 与 CPM 中的元素 e 半匹配,记为 $e'\rightarrow e$.

$(Metaclass(e)=\text{'WorkProduct'})\wedge(Metaclass(e)=Metaclass(e'))\wedge(e.Name\approx e'.Name)\wedge(\exists s(\forall s's'\not\approx s)(s\in State(e)s'\in State(e')))\Rightarrow e'\rightarrow e$.

匹配是很多研究领域需要解决的问题,例如面向 Web 的数据整合、电子商务、数据库设计等. 语言方法(Linguistic Approaches)是一种主要的匹配方法[12]. 本文采用名称匹配(name matching),匹配条件是名称相同. CMM 词库支持判断元素属性是否语义相同或相似,词库中包含 CPM 中的过程元素属性在具体实施时可能用到的名称,如果 EPM 中元素属性可以在其中找到名称相同的项,则说明 CPM 与 EPM 中的过程元素属性语义相同或相近. 对于无法通过词库自动完成的复杂语义判断,需要人工参与,例如在 EPM 中有"项目测试人员"角色,其职责包括承担 CPM 中"质量保证人员"的工作,因此单纯依靠词库无法确定二者之间的关系,必须由用户进行指定.

活动片段匹配基于元素匹配进行,如果元素匹配为半匹配,融合方法将调整 EPM,保证元素匹配为完全匹配,因此活动片段匹配只有完全匹配一种情况.

定义 5. 当且仅当满足下列匹配规则时,CPM 中的活动片段 AS_1 与 EPM 中的活动片段 AS_1' 完全匹配,记为 $AS_1\leftrightarrow AS_1'$.

$(\forall e(\exists e'e'\leftrightarrow e))\wedge(\forall e'(\exists e\ e\leftrightarrow e'))(e\in AS_1,e'\in AS_1')\Rightarrow AS_1\leftrightarrow AS_1'$.

3.1.4 模型间的一致性

过程模型融合要解决的关键问题是模型间的一致性. CPM 中给出 CMM 过程改进的一般性过程,当 EPM 中的元素和关系不符合 CPM 时,均被认为产生了不一致. 由于在实施 CMM 时,允许企业根据自身特点保留某些与 CMM 不一致的企业过程特征,所以当出现这种情况时,不能直接判断该不一致是错误的. 为了更好地解决以上问题,我们将不一致情况分为两类:绝对不一致和相对不一致. 绝对不一致指的是 EPM 中缺乏不可裁剪的 CMM 类型元素或者关系. 绝对不一致的处理可以直接依据 CPM,在 EPM 中添加缺少的 CMM 类型元素或关系. 例如 CPM 中定义了不可裁剪的"评审"活动,但 EPM 中没有定义该类型活动. 通过在 EPM 中添加"评审"活动,可解决这种不一致. 相对不一致的情况比较复杂,解决方式需要用户确认. 相对不一致主要分为 4 种,如表 2 所示.

表 2 相对不一致分类

序号	相对不一致分类	举例
(1)	EPM 中 CMM 类型片段间边 E' 与 CPM 中的匹配片段间边 E 矛盾,则删除 EPM 中的 E',建立 E,并提示用户确认.	CPM 中从"评审"片段到"创建基线"片段存在有向边 E,EPM 中从"创建基线"片段到"评审"片段存在有向边 E',E' 与 E 矛盾,则删除 EPM 中的 E',增加 E,并提示用户确认.

（续　表）

序号	相对不一致分类	举例
(2)	EPM 中两片段间增加了 CPM 未定义的有向边 E'，若该两片段间还存在逆向可达路径 Rx'，且 Rx' 满足 CPM 约束，从而造成环路，需删除 E'，并提示用户确认.	EPM 中从“创建基线”到“编写需求规约”片段之间增加了有向边 E'，而从“编写需求规约”到“创建基线”还存在可达路径 Rx'，且 Rx' 满足 CPM 约束，则删除 E'，并提示用户确认.
(3)	EPM 中两片段间增加了 CPM 未定义的有向边 E'，若在 CPM 中的匹配片段间可达路径上，还存在其它片段 AS_x，AS_x 不可裁剪，且在 EPM 中没有匹配项，则删除 E'，并提示用户确认.	EPM 中从“编写需求规约”片段到“创建基线”片段存在有向边 E'，CPM 中三个片段“编写需求规约”、“评审”、“创建基线”之间存在路径 $R.E'$ 在 CPM 中未定义，且 CPM 中不可裁剪的“评审”片段在 EPM 中没有匹配项. 由于 E' 是在缺少“评审”的条件下定义的，因此应删除 E'，并提示用户进行确认.
(4)	除(2)和(3)外，在满足 CPM 约束的条件下，EPM 中两个 CMM 类型元素或者片段之间增加的企业自定义关系将得到保留，并提示用户确认.	EPM 中角色“高级经理”执行活动“评审项目里程碑工作产品”，该关系在 CPM 中未定义，但由于企业特殊需要，可予以保留.

3.2　过程模型融合方法

过程模型融合的主要操作是匹配和调整，每一次匹配之后，都将根据发现的不一致情况进行模型调整. 在基于模型融合的 CMM 实施过程建模中，过程模型融合依次执行以下 5 个算法.

算法 1.　元素融合.

过程模型融合中，首先进行元素融合，通过元素融合，补充 EPM 缺少的 CPM 中不可裁剪的元素. 在元素融合中，先进行 CPM 与 EPM 元素匹配，如果元素完全匹配成功，则记录匹配对；如果为半匹配，则增加缺少的属性，保证元素完全匹配，并记录匹配对；如果匹配不成功，对于 CPM 中不可裁剪的元素，在 EPM 中生成该元素的匹配项. 元素融合算法如下：

1. 对 CPM 和 EPM 分别进行广度优先搜索，遍历模型中的元素. CPM 中的元素及属性保存在数组 $CpmList$ 中，EPM 中的元素及其属性保存在数组 $EpmList$ 中.

2. 对 $CpmList$ 中的每一项 e，在 CMM 词库中取出 e 的 $Name$ 属性对应的实施元素名称集合 m，依次取出 $EpmList$ 中与 e 同类型的元素 e'：

a. $Metaclass(e)=$('Role'∨'Activity')，如果 e' 的名称与 m 中某一元素名称相同，则 $e\leftrightarrow e'$，记入 $CpmList$，e' 标记为 CMM 类型元素；

b. ($Metaclass(e)=$('WorkProduct'))，在词库中取出 $State(e)$ 中所有状态对应的实施元素名称集合：

i. 如果对于 e 的任一状态 s，其实施元素名称集合为 $n.State(e')$ 中总存在与 n 中某一元素名称相同的状态，则 $e\leftrightarrow e'$，记入 $CpmList$，将 e' 标记为 CMM 类型元素；

ii. 如果对于 e 的某个状态 s，其实施元素集合为 $n.State(e')$ 中不存在与 n 中的任一元素名称相同的状态，则 $e'\rightarrow e$，$addStateToWorkproduct(e.s,e')$，保证 $e'\leftrightarrow e$，填入 $CpmList$，将 e' 标记为 CMM 类型元素.

c. 匹配结果由用户进行确认，并由用户手工指定复杂语义匹配.

3. 对于 $CpmList$ 中未匹配成功的 CPM 中的元素，读取裁剪规则，对可裁剪的元素进行标记. 对不可裁剪的元素 e，如果 EPM 中没有 e 的完全匹配项，则在 EPM 中增加元素 e'，其中 $e'=Clone(e)$，保证 e' 与 e 的所有属性均相同，将 e' 标记为 CMM 类型元素.

4. 将 $EpmList$ 中未匹配成功的元素，标记为自由元素.

算法 2.　活动片段融合.

活动片段融合保证 EPM 中角色、活动和工作产品之间的关系满足 CPM. 活动片段融合算法如下：

1. 按照活动片段分解 CPM(除去可裁剪的且未匹配成功的元素)，将活动片段放入数组 $CpmSegment$ 中.

2. 在 $CpmSegment$ 中依次取活动片段 $AS:=(A,R,W)$，此时 A，R 和 W 中的所有元素在 EPM 中均已有完全匹配项. 假设 A 的匹配项为 A'(A' 为 A 匹配项集合中的元素).

3. 任一角色 $R_i(R_i\in R)$ 与活动 A 的关系为 R_i 执行 A，可表示为 $L(R_i,A)$. 假设 R_i 的匹配项为 $R_i'(R_i'$ 为 R_i 匹配项集合中的元素)：

a. 如果$(\forall R_i'\ \exists A'\ L(R_i',A'))\wedge(\forall A'\ \exists R_i'L(R_i',A'))$，则该关系匹配成功.

b. 如果$(\exists R_i'\ \forall A'\ \neg L(R_i',A'))\vee(\exists A'\ \forall R_i'\neg L(R_i',A'))$，则提示用户指定 $L(R_i',A')$.

4. 活动 A 与任一工作产品 $W_i(W_i\in W)$ 的关系为 A 输出 $W_i\langle s\rangle$，表示为 $L(A,W_i\langle s\rangle)$，假设 $W_i\langle s\rangle$ 的匹配项为 $W_i'\langle s'\rangle(W_i'$ 为 W_i 的匹配项集合中的元素，s' 为 s 的匹配项集合中的元素)：

a. 如果 $\forall W_i'\langle s'\rangle(\exists A'\ L(A',W_i'\langle s'\rangle))\wedge(\forall A_x'\,L(A_x',W_i'\langle s'\rangle)\wedge(A_x'\leftrightarrow A)\wedge(A_x'=A'))\wedge(\forall A'\ \exists W_i'\langle s'\rangle L(A',W_i'\langle s'\rangle))$，则该关系匹配成功.

b. 如果$(\exists A'\ \forall W_i'\langle s'\rangle\neg L(A',W_i'\langle s'\rangle))$，则提示用户在没有相应输出活动的工作产品中选择 $W_i'\langle s'\rangle$，建立 $L(A',W_i'\langle s_i'\rangle)$. 如果$(\exists W_i'\langle s'\rangle\forall A'\ \neg L(A',W_i'\langle s'\rangle))$，提示用户仅选择一个 A'，建立 $L(A',W_i'\langle s_i'\rangle)$. 在此基础上，如果 EPM 中还存在 $L(A'',W_i'\langle s'\rangle)$，且 A'' 不是 A 的匹配项，则删除 $L(A'',W_i'\langle s'\rangle)$.

5. 当所有关系都匹配完毕，由用户确认. 活动片段 AS 中所有元素的匹配项组成活动片段，这些片段可以组成 AS 的匹配项集合 AS'，将 AS' 记入 $CpmSegment$ 数组.

6. 返回步 2，直至 $CpmSegment$ 中所有的活动片段都匹配结束.

7. 生成带标记的有向图 CPM_G，EPM_G. 为方便论述，将活动片段 AS 对应的顶点记为 V_{AS}：

a. 依据 CPM，生成 CPM_G，活动片段抽象为顶点，边标记为 HasV. 这是因为 CMM 在实施过程中，允许企业在制定的 CMM 实施过程中添加体现企业过程特征的元素，因此 CPM_G 中的边均标记为 HasV，表示两个顶点间可以加入其它顶点. 如果活动片段 AS_2 中的活动可裁剪，其前后相连的活动片段分别为 AS_1，AS_3. 建立 $E(V_{AS_1}, V_{AS_3})$，标记为 $Exist$，表示 V_{AS_1}，V_{AS_3} 之间可存在有向边.

b. 依据 EPM，生成 EPM_G. 活动片段抽象为顶点，其中的自由片段对应的顶点标记为 F，记录二者对应关系. 边的标记均为 NoV. 如果 CPM 中的 AS_x 与 EPM 中的 AS_a，AS_b 存在匹配关系：$AS_x \leftrightarrow \{AS_a, AS_b\}$，将 V_{AS_a}，V_{AS_b} 合并为一个顶点 V'_{AS_x}，将 V_{AS_a}，V_{AS_b} 的关联边全部加到 V'_{AS_x} 上，记入数组 $UnionV$.

算法 3. EPM 有向图剪枝.

由于 EPM_G 中包含 F 顶点，为 EPM_G 和 CPM_G 的融合操作带来一定难度. 我们通过对 EPM_G 进行剪枝，去掉其中的 F 顶点，增强模型融合可操作性. EPM_G 剪枝规则如下：

1. 如果 AS_1，AS_2，AS_3 均为 CMM 类型片段. 路径 R 表示为 $(V_1, V_2, \cdots, V_n)$，其中的顶点标记均为 F，则 R 称为自由路径. 对 EPM_G 进行广度搜索，若 V_{AS_1} 与 R 中的 V_1 相连，且 $V_2, V_3, \cdots, V_n$ 均不与其它 CMM 类型片段所对应的顶点相连，则将 R 删除，把 V_{AS_1} 和 R 记入数组 $Delete$，其中 V_{AS_1} 作为 R 的相连元素标记.

2. 如果 V_{AS_1} 与 V_{AS_2} 之间存在 R，但 R 中存在顶点 V_x 与 V_{AS_3} 相连，则建立 $E(V_{AS_1}, V_{AS_2})$，$E(V_{AS_1}, V_{AS_3})$，均标记为 HasV，将 R 标记为删除. 把 R 记入数组 $Delete$，对 V_{AS_1}，V_{AS_2}，V_{AS_3}，V_x 也进行记录.

3. 当 EPM 中所有顶点都搜索完毕，将标记为删除的自由路径删掉. 得到 EPM'_G.

如图 4，EPM_G 中顶点 1′，2′，3′和 4′代表 CMM 类型片段，F 代表自由片段. 剪枝后，EPM'_G 中只留下 1′，2′，3′，4′及其边.

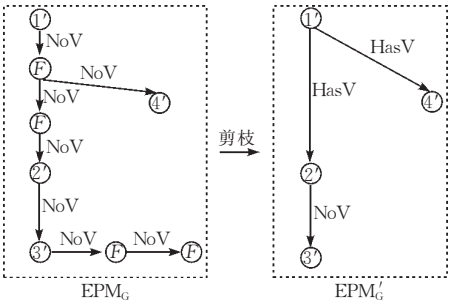


图 4 EPM 有向图剪枝

算法 4. 路径融合.

EPM 过程元素顺序关系与 CPM 不一致是模型融合要解决的重要问题，通过路径融合可以保证最后得到的 CMM 类型片段间顺序关系满足 CPM 要求. 路径融合算法如下：

1. 使用广度优先搜索方式取 CPM_G 中的边，假设为 $E(V_1, V_2)$，读取其标记 L，在 EPM'_G 中搜索 V_1，V_2 的对应顶点 V'_1，V'_2，判断是否存在 $E(V'_1, V'_2)$：

a. 如果 $E(V'_1, V'_2)$ 存在，则匹配成功，保留 $E(V'_1, V'_2)$ 及其标记，并记录 $E(V'_1, V'_2)$ 为已读；

b. 否则：

i. 建立 $E(V'_1, V'_2)$，如果 L 为 Exist，则设 $E(V'_1, V'_2)$ 的标记为 Exist；否则设为 NoV. 记录 $E(V'_1, V'_2)$ 为已读.

ii. 如果 $E(V'_2, V'_1)$ 存在，则删除 $E(V'_2, V'_1)$，并提示用户确认. 转到步 2.

2. 回到步 1，直到 CPM_G 中所有的边都匹配完毕，转到步 3.

3. 使用广度优先搜索方式依次取 EPM'_G 中未读的边，假设为 $E(V'_1, V'_2)$，在 CPM_G 中搜索其对应顶点 V_1 和 V_2. 如果 $E(V_1, V_2)$ 不存在：

a. 如果存在 $R(V'_2, V'_1)$，造成环路，则删除 $E(V'_1, V'_2)$，并提示用户确认. 转到步 4.

b. 否则，

i. 如果除 $E(V'_1, V'_2)$ 之外，EPM'_G 中不存在其它的 V'_1 到 V'_2 的路径，或者 V'_1 到 V'_2 的路径经过的所有顶点对应的片段中活动元素均不是在元素融合时添加的，则保留 $E(V'_1, V'_2)$ 及其标记，并提示用户进行确认. 将 $E(V'_1, V'_2)$ 标记为已读，转到步 4.

ii. 如果除 $E(V'_1, V'_2)$ 之外，EPM'_G 中存在其它 V'_1 到 V'_2 的路径 (V'_1, V'_x, V'_2)，并且其中 V'_x 所对应的活动片段中的活动为元素融合时增加的，则删除 $E(V'_1, V'_2)$，并提示用户确认. 转到步 4.

4. 如果 EPM'_G 中还有未读边，则转到步 3；否则，路径融合完毕，将调整后的 EPM'_G 保存为 $CIPM_G$.

如图 5 所示，按照路径融合算法，CPM_G 与 EPM'_G 进行融合，得到 $CIPM_G$. 其中 EPM'_G 的顶点 4′对应的活动片段中的活动是在元素融合时新增加的. 对于 $E(1,2)$，EPM'_G 中 $E(1', 2')$ 与之对应，保留 $E(1', 2')$，其标记仍为 NoV；对于 $E(1,3)$，其标记为 Exist，EPM'_G 中缺少其对应边，则建立 $E(1', 3')$，标记为 Exist；对于 $E(2,3)$，在 EPM'_G 中只存在逆向边 $E(3', 2')$，说明顶点 2′与顶点 3′的顺序关系与 CPM_G 冲突，需要删除 $E(3', 2')$，建立 $E(2', 3')$，其标记设为 $E(3', 2')$ 原有标记 NoV，并要求用户确认；对于 $E(2,4)$ 和 $E(4,5)$，EPM'_G 中缺少其对应边，增加 $E(2', 4')$ 和 $E(4', 5')$，标记设为 NoV；对于 EPM'_G 中的 $E(2', 5')$，CPM_G 中没有 $E(2,5)$，且存在路径 $(2,4,5)$，其中 4′属于新增片段，因此删除 $E(2', 5')$，并提示用户确认.

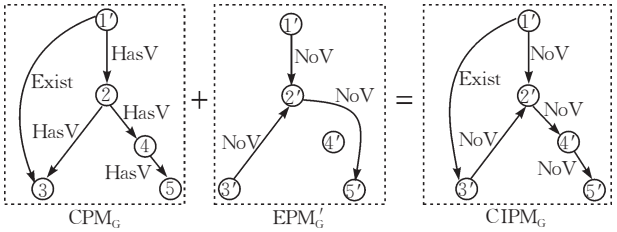


图 5　路径融合

算法 5.　CIPM 恢复.

最后一步是对 $CIPM_G$ 进行扩充，得到完整的 CIPM 过程模型.

1. 对 $CIPM_G$ 进行顶点 F 填充. 读取 EPM_G 剪枝中的数组 $Delete$，将顶点 F 填充到 $CIPM_G$ 中.

2. 读取活动片段融合中的顶点合并记录 $UnionV$，分解顶点，恢复原有的活动片段对应的顶点及路径.

3. 恢复活动片段. 将 $CIPM_G$ 中的顶点和边用 SPEM 中的元素表示. 将 $CIPM_G$ 中的顶点恢复为活动及其相关角色和工作产品，如果两个活动片段具有相同的元素，则将该元素合并为一个. 将 Exist 边注释为“可裁剪”. 最终得到 CIPM.

需要说明的是，由于模型融合过程中需要用户参与，因此如果基于相同的 EPM 进行融合，最终得到的 CIPM 仍然会由于用户选择的不同而有所差异，即 CIPM 并不是绝对唯一的.

3.3　模型融合一致性证明

经过模型融合之后得到的 CMM 实施过程模型 CIPM 满足 2.2.3 节中的一致性约束，下面对 CIPM 满足以上一致性约束的证明基于假定 1.

假定 1.　CPM 和 EPM 均满足 SPEM 元模型约束.

如第 2 节所述，CPM 和 EPM 均基于 SPEM 定义，因此模型融合的前提是 CPM 和 EPM 均是合法的 SPEM 模型，满足 SPEM 元模型约束. 模型融合算法是针对 SPEM 过程元模型设计的，只有在假定 1 成立的情况下，才能得到正确的 CIPM.

引理 1.　CIPM 中所有过程元素及其关系均满足 SPEM 元模型约束.

证明.　首先证明 CIPM 中所有过程元素均是 SPEM 中元类的实例. 基于假定 1，CPM 和 EPM 均满足 SPEM 的约束，由于 CPM 和 EPM 中仅包含 Role、Activity 和 WorkProduct 等元类的实例，且融合算法仅基于 CPM 和 EPM 中的元素进行操作，不增加 SPEM 中元类以外的元素定义，因此可以保证融合后的 CIPM 模型中的元素均是 SPEM 中元类的实例.

然后证明 CIPM 中过程元素间关系满足 2.1 节中 SPEM 元模型的下列约束.

(1)所有元素的名称均唯一.

基于假定 1，可知 CPM 和 EPM 中所有元素的名称均唯一. 由于模型融合是依据 CPM 对 EPM 进行调整的过程，因此仅需要证明 EPM 在调整过程中增加的元素名称是唯一的且不与 EPM 中原有元素名称重复. 融合方法中仅有算法 1 会在 EPM 中增加新元素，这些元素是 EPM 缺少的、CPM 中不可裁剪的元素. 算法 1 采用的匹配方式是针对相同类型元素进行名称匹配，因此如果 CPM 中不可裁剪的元素 $e_1, e_2, \cdots, e_n$ 在 EPM 中没有匹配项，则说明 EPM 中同类型元素的名称均不与任意 e_i ($e_i \in \{e_1, e_2, \cdots, e_n\}$) 相同，算法 1 会在 EPM 中增加 e_i 的匹配项 e_i'，且 $e_i'.Name = e_i.Name$，因此 $e_i'.Name$ 不会与 EPM 中的其它元素重复，并且由于 $e_1, e_2, \cdots, e_n$ 的名称均不相同，则 $e_1', e_2', \cdots, e_n'$ 的名称也不相同.

(2)一个具有特定状态的工作产品只能由一个活动输出，一个活动至少输出一个具有特定状态的工作产品.

由假定 1 可知，EPM 中的过程元素关系满足 SPEM 约束. 算法 1 会在 EPM 中增加其缺少的 CPM 中不可裁剪的元素，这些元素间关系在算法 2 中进行定义，因此仅需要证明算法 2 的操作满足约束(2). 如果 $W_i'\langle s_i'\rangle$ 不由任何活动 A_i' 输出 ($A_i' \in \{A_1', A_2', \cdots, A_n'\}$，$\{A_1', A_2', \cdots, A_n'\}$ 为 CPM 中 A 的匹配项集合)，算法 2 会要求用户仅选择一个 A_i'，并建立 $L(A_i', W_i'\langle s_i'\rangle)$；如果活动 A_i' ($A_i' \in \{A_1', A_2', \cdots, A_n'\}$) 没有输出任何 $W_i'\langle s_i'\rangle$ ($W_i' \in \{W_1', W_2', \cdots, W_n'\}$，$s_i' \in \{s_1', s_2', \cdots, s_n'\}$，$\{W_1', W_2', \cdots, W_n'\}$ 为 CPM 中 W 的匹配项集合，$\{s_1', s_2', \cdots, s_n'\}$ 为 W 的属性 s 的匹配项集合)，则算法 2 会要求用户选择没有相关输出活动 A_j' ($A_j' \in \{A_1', A_2', \cdots, A_n'\}$) 的 $W_i'\langle s_i'\rangle$，并建立 $L(A_i', W_i'\langle s_i'\rangle)$. 对于以上两种情况，如果还存在 $L(A'', W_i'\langle s_i'\rangle)$，其中 $A'' \notin \{A_1', A_2', \cdots, A_n'\}$，则算法 2 会删除 $L(A'', W_i'\langle s_i'\rangle)$，保证 $W_i'\langle s_i'\rangle$ 最终只由一个活动输出.

(3)一个活动至少由一个角色执行，一个角色至少执行一个活动；

同约束(2)的证明，仅需要证明算法 2 的操作满足约束(3). 如果活动 A_i' ($A_i' \in \{A_1', A_2', \cdots, A_n'\}$) 没有相应的执行角色或者 EPM 中角色 R_i' ($R_i' \in \{R_1', R_2', \cdots, R_n'\}$) 没有执行活动，算法 2 会要求用户建立 $L(R_i', A_i')$，保证每个活动至少有一个执行角色，每个角色至少有一个活动执行.　　证毕.

引理 2.　CIPM 中保留 CPM 和 EPM 中必要

的过程元素.

证明. 首先证明 CPM 中任何不可裁剪的元素 e 在 CIPM 中均得到保留. 元素 e 在 EPM 中的匹配情况分为:EPM 中存在其匹配项 e'、EPM 中不存在其匹配项. 对于第一种情况,e' 是 CMM 类型元素,在融合过程中不会被删除,因此 e 在 CIPM 中得到保留,对于第二种情况,算法 1 会在 EPM 中生成其匹配项 e',同样由于 e' 是 CMM 类型元素,在后续的融合过程中不会被删除,将保留在 CIPM 中. 因此 CPM 中任何不可裁剪的 CMM 元素在 CIPM 中均得到保留.

然后证明对于 CPM 中的过程元素 e,如果 e 为可裁剪元素且在 EPM 中没有匹配项,则 e 在 CIPM 中不会得到保留. 在算法 1 中,如果在 EPM 中找不到 e 的匹配项,算法 1 不会对 EPM 进行任何操作,因此 e 不会在 CIPM 中得到保留.

最后证明 EPM 中所有过程元素在 CIPM 中均得到保留. 模型融合中对 EPM 中过程元素的删除操作只发生在算法 3,其中会将 EPM 中的自由元素进行删除,并保存在数组 *Delete* 中. 在算法 5 中,*Delete* 记录中所有被删除元素均将在 CIPM 中进行恢复,即算法 3 所删除的自由元素会重新填充到 CIPM 中,因此 EPM 中所有过程元素在 CIPM 中均得到保留. 证毕.

引理 3. CIPM 中的 CMM 类型过程元素关系与 CPM 一致.

证明. 要证明 CIPM 中 CMM 类型过程元素关系与 CPM 一致,必须证明 CIPM 中既不存在绝对不一致也不存在相对不一致. 下面首先证明 CIPM 中不存在绝对不一致. 如果 CIPM 中缺少 CPM 中的不可裁剪元素或者关系,则说明存在绝对不一致. 由引理 2 可知,CPM 中任何不可裁剪元素在 CIPM 中得到保留,因此接下来只需要证明 CIPM 中包含所有的不可裁剪元素间关系. 过程元素关系分为两种:(1)角色、活动和工作产品(状态)之间的关系;(2)活动片段之间的关系. 算法 2 处理的是第一类关系,角色与活动之间的执行关系以及活动与工作产品(状态)之间的输出关系均依据 CPM 进行判断,如果执行关系或输出关系缺少,算法 2 均会依据 CPM 进行建立,并且由引理 1 可知,所建立的关系满足 SPEM 元模型的约束. 算法 4 处理的是第二类关系. 如果 EPM 中两个不可裁剪片段间缺少 CPM 中定义的边 E,则算法 4 会建立 E',同时消除存在的环路. 因此 CIPM 中包含 CPM 中所有不可裁剪元素及其关系,不存在绝对不一致的情况.

然后证明 CIPM 中不存在相对不一致. 相对不一致共有 4 类,见表 2. 对于第(1)类情况,如果 CPM 片段间存在边 $E(V_1,V_2)$,EPM 对应片段间存在 $E(V_2',V_1')$,算法 4 会删除 $E(V_2',V_1')$,建立 $E(V_1',V_2')$,保证 EPM 中片段间关系与 CPM 一致. 对于第(2)类情况,EPM 中两片段间存在关系 $E(V_1',V_2')$,同时存在 $R(V_2',V_1')$,如果 $R(V_2',V_1')$ 与 CPM 一致,算法 4 会删除 $E(V_1',V_2')$. 对于第(3)类情况,EPM 中两片段间存在 $E(V_1',V_2')$,在 CPM 的对应片段间存在 $R(V_1,V_x,V_2)$,且 V_x 不可裁剪并在 EPM 中没有匹配项,算法 4 会删除 $E(V_1',V_2')$. 对于第(4)类情况,在满足 CPM 约束的情况下,算法 2 和算法 4 中会保留 EPM 中具有企业特色的过程元素关系. 因此,CIPM 中所有相对不一致均会在融合方法中得到处理,CIPM 不存在相对不一致的情况. 证毕.

引理 4. CIPM 中的自由元素间的关系与 EPM 一致.

证明. CIPM 中的自由元素来自 EPM,自由元素及其关系不受 CPM 约束. 在算法 3 中,为了便于模型融合,EPM 中的自由元素均被删除,这些被删除的自由元素间的关系及其相关的 CMM 类型元素均被记录下来,并在算法 5 中按照原有记录重新填充进入 CIPM,因此自由元素间的关系仍然与 EPM 一致. 证毕.

引理 5. CIPM 中不包含 CPM、EPM 以外的过程元素和关系.

证明. 首先证明 CIPM 中不包含 CPM、EPM 以外的过程元素. 假设 CIPM 中包含 CPM、EPM 以外的过程元素,则该元素必是在融合过程中由某个算法引入的. 算法 1 将 EPM 中缺少的 CPM 中不可裁剪的元素添加到 EPM 中,所添加的是 CPM 中元素的匹配项,不是 CPM 以外的元素;算法 2 和算法 4 进行 EPM 中元素关系的处理,不涉及元素的增加,因此不会引入其它元素;算法 3 将 EPM 中的自由元素进行删除,也不会引入其它元素;算法 5 将算法 3 中删除的自由元素重新添加到 EPM 中,这些自由元素是 EPM 原有的,不会引入其它元素. 因此该假设不成立,CIPM 中不包含 CPM、EPM 以外的过程元素.

下面证明 CIPM 中不包含 CPM、EPM 以外的过程元素关系. 证明方式同上,算法 1 所作的操作是在 EPM 中添加元素,不涉及元素关系;算法 2 和算

法 4 仅增加 CPM 中定义的但 EPM 中缺少的关系，因此不会引入其它关系；算法 3 仅删除自由元素及其关系；算法 5 所增加的关系是 EPM 中原有的. 因此模型融合不会引入除 CPM、EPM 以外的元素和关系.　　证毕.

定理 1.　融合后的 CIPM 满足模型一致性约束.

证明.　根据引理 1、引理 2、引理 3、引理 4 和引理 5，可知 CIPM 中所有过程元素及其关系均满足 SPEM 元模型约束，保留了 CPM 和 EPM 中必要的过程元素，其中的 CMM 类型过程元素关系与 CPM 一致，且自由元素间的关系与 EPM 一致，最后 CIPM 中不包含 CPM、EPM 以外的过程元素和关系. 因此 CIPM 满足模型一致性约束. 定理 1 成立.　　证毕.

4　工具及实例

4.1　过程模型融合原型工具

基于以上融合方法，我们实现了过程模型融合原型工具，如图 6 所示. 该原型工具支持基于 SPEM 建立 CMM 过程模型 CPM 和企业过程模型 EPM，并实现模型融合机制，最终得到 CMM 实施过程模型 CIPM. 在模型融合的过程中，工具将发现的不一致问题以及调整操作以提示信息的形式显示给用户. 对于复杂匹配以及相对不一致问题的解决，将要求用户进行确认.

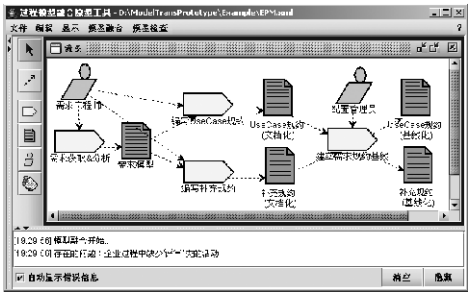


图 6　过程模型融合原型工具

4.2　一个实例

为验证所提出的融合方法，我们在项目实践中进行了应用，下面选择某组织的需求管理过程为例进行说明. 首先建立两个需求管理过程模型：CMM 需求管理过程模型 CPM_RM 和企业需求管理过程模型 EPM_RM，如图 7 所示. CPM_RM 中需求规约形成文档之后，由项目经理及相关组等角色进行评审，评审通过之后，交给配置管理员创建基线. EPM_RM 中需求工程师首先进行需求获取 & 分析，建立需求模型，然后编写 UseCase 规约和补充规约，最后将成文的需求规约直接交给配置管理员创建基线. 可以看出，EPM_RM 中缺少不可裁剪元素“评审”及其相关角色和工作产品，下面利用模型融合方法对 EPM_RM 进行调整：

(1)元素融合. CPM_RM 与 EPM_RM 间元素的匹配关系如表 3 所示. 表中 EPM_RM 列的白色填充项为 CPM_RM 与 EPM_RM 原有的匹配项. 灰色填充项为元素融合过程中融合工具在 EPM_RM 中新加入的元素或者属性，如增加了活动“评审”，角色“相关组”和“项目经理”，工作产品“UseCase 规约”和“补充规约”均增加了状态“〈已评审〉”. EPM_RM 中除活动“需求获取与分析”及工作产品“需求模型”为自由元素以外，其它均为 CMM 类型元素.

表 3　元素匹配关系

元类		CPM_RM		EPM_RM	
活动		编写规约		编写 UseCase 规约	
				编写补充规约	
		创建基线		建立需求规约基线	
		评审		评审	
工作产品	〈状态〉	需求规约	〈文档化〉	UseCase 规约	〈文档化〉
			〈已评审〉		〈已评审〉
			〈基线化〉	补充规约	〈基线化〉
角色		规约编写者		需求工程师	
		配置管理员		配置管理员	
		相关组		相关组	
		项目经理		项目经理	

(2)活动片段融合. CPM_RM 中有 3 个活动片段：AS_1＝{(编写规约)，(规约编写者)，(需求规约〈文档化〉)}，AS_2＝{(评审)，(相关组，项目经理)，(需求规约〈已评审〉)}，AS_3＝{(创建基线)，(配置管理员)，(需求规约〈基线化〉)}. EPM_RM 中有 4 个活动片段：AS_a＝{(需求获取 & 分析)，(需求工程师)，(需求模型〈NULL〉)}，AS_b＝{(编写 UseCase 规约)，(需求工程师)，(UseCase 规约〈文档化〉)}，AS_c＝{(编写补充规约)，(需求工程师)，(补充规约〈文档化〉)}，AS_d＝{(建立需求规约基线)，(配置管理员)，(UseCase 规约〈基线化〉，补充规约〈基线化〉)}.

其中 $AS_1 \leftrightarrow \{AS_b, AS_c\}$，$AS_3 \leftrightarrow AS_d$，$AS_a$ 为自由片段. AS_2 在 EPM_RM 中没有匹配项，通过活动片段融合，依据 AS_2，在 EPM_RM 中建立 AS_2 元素的匹配项间关系，得到活动片段 AS_e＝{(评审)，(相关组，项目经理)，(UseCase 规约〈已评审〉，补充规约〈已评审〉)}，记为 $AS_2 \leftrightarrow AS_e$. 如图 7 所示，生成有向图 CPM_RM_G 和 EPM_RM_G. CPM_RM_G 中顶

点 1,2,3 分别代表 AS_1,AS_2,AS_3. EPM_RM_G 中顶点 F 代表 AS_a,顶点 1′代表 AS_b,AS_c,顶点 2′代表 AS_e,顶点 3′代表 AS_d.

(3)EPM 有向图剪枝. 对 EPM_RM_G 进行剪枝,去掉顶点 F,得到的 $EPM\_RM'_G$,如图 7 所示.

(4)路径融合. 路径融合中,依据 CPM_RM_G,建立 $E(1',2')$,$E(2',3')$,设标记为 NoV. 对 $E(1',3')$,CPM_RM_G 中不存在 $E(1,3)$,按照路径融合算法,由于 2′对应的片段中的活动"评审"是元素融合时增加的,即 $E(1',3')$ 是在缺少"评审"的条件下定义的,因此删除 $E(1',3')$,并提示用户进行确认. 如图 7 所示,得到 $CIPM\_RM_G$.

(5)CIPM_RM 恢复. 将顶点 F 填充到 $CIPM\_RM_G$ 中,合并的顶点 1′分解为两个顶点,分别对应 AS_b,AS_c. 恢复所有片段中的元素,将重复的元素合并,得到最终的 CIPM_RM 模型,如图 7 所示. 可以看出 CIPM_RM 中已增加了评审活动,并保留了 EPM_RM 自身的过程特征.

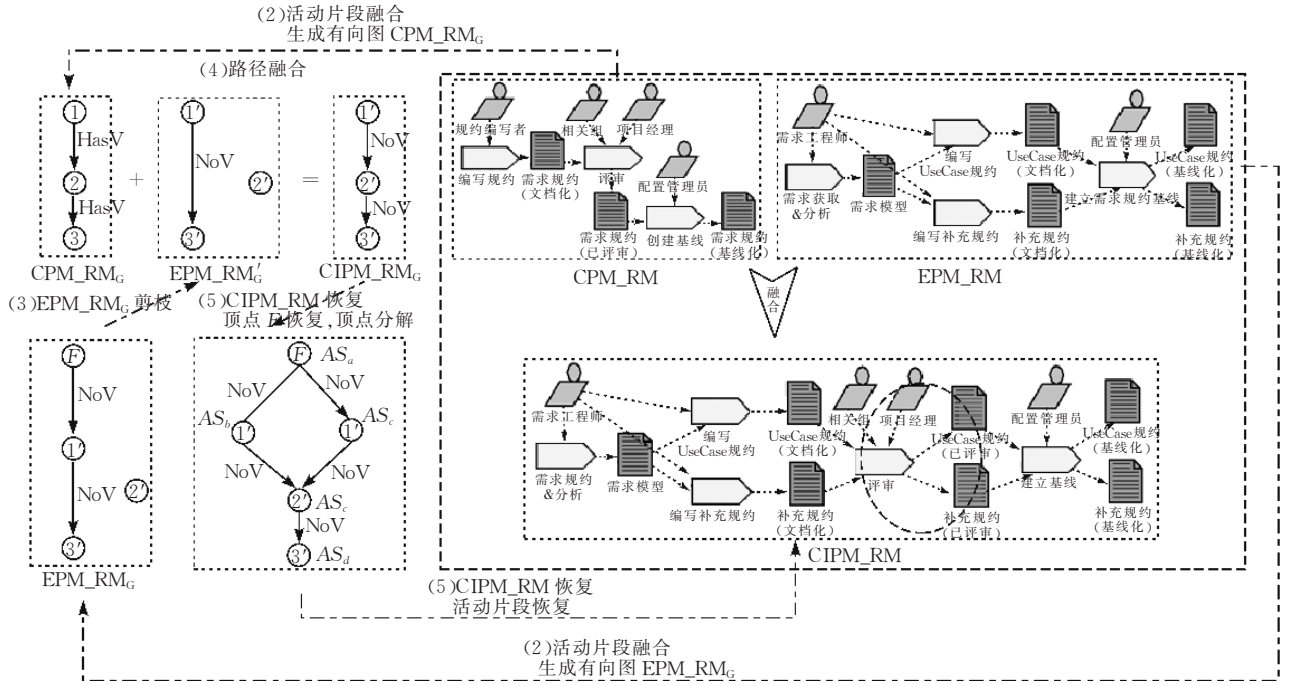


图 7 CPM_RM 与 EPM_RM 融合

5 结 语

在软件过程改进中,由于企业过程的复杂性和易变性,直接进行 CMM 实施过程建模将极大地增加 CMM 过程改进的成本和风险. 针对此问题,本文提出一种基于模型融合的 CMM 实施过程建模方法. 该方法易于操作,可以有效地支持 CMM 过程改进,当企业业务过程发生改变时,利用模型融合机制可以及时地对 CMM 实施过程模型进行修改,从而提高 CMM 过程改进的效率和质量,并保证模型的正确性. 方法中使用 SPEM 作为过程模型建模的元模型,分别建立 CMM 过程模型 CPM 和企业过程模型 EPM,通过 CPM 和 EPM 的模型融合来建立 CMM 实施过程模型 CIPM. 该方法使用带标记的有向图来描述过程模型,根据过程模型融合的目标,通过元素融合、活动片段融合、EPM 有向图剪枝、路径融合和 CIPM 恢复等算法,最终得到 CIPM. 文中对模型融合满足一致性约束进行了证明并给出一个支持该方法的过程模型融合原型工具及相关实例.

2002 年,SEI 综合不同领域的 CMM,提出 CMMI (Capability Maturity Model Integration) 模型[13]. 由于 CMMI 模型以 CMM 为基础,并具备过程特点,本文的融合方法经过扩展同样可以应用于 CMMI 模型与企业过程模型的模型融合,这也是有待进一步研究的问题.

致 谢 在此,我们向对本文工作给予支持和建议的赵欣培、武占春、徐伟、汪锦岭和刘绍华五位同学表示感谢!

参 考 文 献

1 Berstein P. A., Halevy A. Y., Pottinger R. A.. A vision for

management of complex models. ACM SIGMOD Record, 2000, 29(4): 55～63

2 Pottinger R. A., Bernstein P. A.. Merging models based on given correspondences. In: Proceedings of the 29th International Conference on Very Large Databases (VLDB) Conferences, Berlin, 2003, 862～873

3 Buneman P., Davidson S., Kosky A.. Theoretical aspects of schema merging. In: Proceedings of the 3rd International Conference on Extending Database Technology: Advances in Database Technology(EDBT), Vienna, Austria, 1992, 152～167

4 Noy N. F., Musen M. A.. PROMPT: Algorithm and tool for automated ontology merging and alignment. In: Proceedings of the 7th National Conference on Artificial Intelligence and the 12th Conference on Innovative Applications of Artificial Intelligence, Austin, Texas, USA, 2000, 450～455

5 Egyed A. F.. Heterogeneous view integration and its automation[Ph. D. dissertation]. University of Southern, California, 2000, 1～249

6 Mens T.. A state-of-the-art survey on software merging. IEEE Transactions on Software Engineering, 2002, 28(5): 449～462

7 Alanen M., Porres I.. Difference and union of models. In: Stevens P. *et al*. eds.. The Unified Modeling Language: Modeling Languages and Applications, Lecture Notes in Computer Science 2863, Springer, 2003, 2～17

8 Object Management Group. Software process engineering metamodel specification, Version 1.0. OMG Document, formal/02-11-14, Needham: Object Management Group, 2002. http://www.omg.org/cgi-bin/doc?formal/2002-11-14

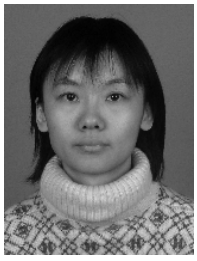
9 Américo S., Alexandre V., Falcone Sampaio P. R.. XWebProcess: Agile software development for web applications. In: Proceedings of the 4th International Conference on Web Engineering (ICWE 2004), 2004, 597～598

10 Object Management Group, UML 2.0 OCL final adopted specification(ptc/03-10-14): Needham, MA, Object Management Group, 2003. http://www.omg.org/cgi-bin/doc?ptc/2003-10-14

11 Li Juan, Li Ming-Shu, Wu Zhan-Chun, Wang Qing. A SPEM-based software process metamodel for CMM. Journal of Software, 2005, 16(8): 1366～1377(in Chinese)
(**李　娟,李明树,武占春,王　青. 一个基于 SPEM 的 CMM 软件过程元模型. 软件学报**, 2005, 16(8): 1366～1377)

12 Rahm E., Bernstein P. A.. A survey of approaches to automatic sechma matching. VLDB Journal, 2001, 10(4): 334～350

13 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



LI Juan, born in 1977, Ph. D. candidate. Her research interests include software process improvement, model driven architecture and requirements engineering.

YUAN Feng, born in 1977, Ph. D. candidate. His research interests include software process modeling and model driven architecture.

LI Ming-Shu, born in 1966, Ph. D., professor. His research interests include requirements engineering and real-time systems.

WANG Qing, born in 1964, Ph. D., professor. Her research interests include software quality management, process modeling, knowledge management and real-time systems.

Background

This research belongs to the National High Technology Research and Development Program (863 Program) under grant No. 2002AA116060 and No. 2001AA113080, and is also supported by the National Natural Science Foundation of China under grant No. 60273026 and No. 60473060. In the past years, authors have gained many achievements in the domains of software process improvement and software process modeling. They have published a lot of high-quality papers in journals and conferences in these research domains. And they also have developed a series of products such as CMM-based Software Quality Management Platform. To reduce the complexity of CMM implementation process modeling, the research of this paper concentrates on using model merging technology to model the CMM implementation process.