基于EPMM的软件过程行为偏离诊断研究

朱 锐[1] 李 彤[1,2] 莫 启[1] 张 璇[1] 王一荃[1] 林雷蕾[1] 代 飞[1,2]
(云南大学软件学院 昆明 650091)[1] (云南省软件工程重点实验室(云南大学) 昆明 650091)[2]

摘 要 近年来,随着对PSEE认识的深入,人们逐渐发现实际观察到的过程执行往往和实施的过程模型之间存在一定的偏离(deviation),从而导致PSEE对于实际软件开发活动失去了指导意义。针对软件过程偏离问题,以软件演化过程元模型(EPMM)为基础,在软件过程偏离发现方面,借鉴进程代数的弱互模拟思想,提出过程行为空间表达式,用以构造软件过程的行为空间来检测过程偏离;在软件过程偏离处理方面,提出过程偏离类型的划分及偏离处理策略。这种方法能够发现软件过程实施中普遍存在的过程偏离问题并加以处理来改进软件过程,最终提高软件产品质量。

关键词 软件过程,行为偏离诊断,软件演化过程元模型,软件过程行为空间,进程代数

中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.11.012

Research on Deviation Diagnostic of Software Process Behavior Based on EPMM Modelling

ZHU Rui[1] LI Tong[1,2] MO Qi[1] ZHANG Xuan[1] WANG Yi-quan[1] LIN Lei-lei[1] DAI Fei[1,2]
(School of Software, Yunnan University, Kunming 650091, China)[1]
(Key Laboratory in Software Engineering of Yunnan Province (Yunnan University), Kunming 650091, China)[2]

Abstract In recent years, with in-depth understanding of PSEE, people gradually discover that there are certain deviations between the enacting process model and the actual observed process, which leads to the PSEE loses guiding significance for the actual software development activities. For the process deviation problem, based on the software evolution process meta-model (EPMM), on the side of software process deviation detecting, this paper proposed the process behavior space expression on learning process algebra weak bi-simulation ideas in order to detect the process deviation. On the side of software process deviation handling, the process deviation division of type and handling strategy were given. This method can effectively find prevalent deviations problems in the software process implementation, and improve the software process through dealing with the deviation problems, ultimately improve the quality of software products.

Keywords Software process, Behavior deviation diagnostic, EPMM, Software process behavior space, Process algebra

1 引言

1987年,Osterwil提出"软件过程也是软件"。他在文献[1]中详细阐述了过程的本质以及过程也能够像软件一样执行(Enactment)。目前,这一思想也逐渐被业界广泛接受,人们越来越意识到:软件产品的质量在很大程度上依赖于产品开发时所使用的过程[2,3]。而一个高质量的软件过程也必须是一个持续不断改进的过程[4]。对于软件过程改进的研究,主要分为两类[5]:第一类为软件工业界所关注的、以能力成熟度集成模型(Capability Maturity Model Integration, CMMI[3])为代表的软件过程评估(Assessment)和改进模型;第二类是学术界所关注的软件过程建模(Software Process Modeling),它主要是通过特定的方法对软件过程进行抽象、表示和分析以增加对软件过程的理解,并通过直接或者间接的方式指导实际的软件开发活动[6]。软件过程建模方法的研究主要是围绕着过程建模语言和以过程为中心的软件工程环境(Process-centered Software Engineering Environment, PSEE)[5]展开的。但是,随着人们对PSEE认识的深入,研究人员和实践者也已经意识到一个过程模型要被执行,必须足够的灵活来允许过程改变去面对没有预期的情况[6]。这就是说,实际观察到的过程执行往往和实施的过程模型之间存在一定的偏离(deviation),这就导致PSEE对于实际软件开发活动失去了指导意义。因此,本文以EPMM为基础,借鉴进程代数的弱互模拟思想,提出一种软件过程偏离的诊断方法,以对实施过程模型进行偏离发现,并以此来对偏离进行处理。

软件过程(Software Process)是指用于开发和维护软件

到稿日期:2013-09-26 返修日期:2013-11-16 本文受国家自然科学基金项目(60963007,61262024,61262025),云南省自然科学基金项目(2007F008M),云南大学软件学院学科建设基金资助项目(2010KS01),云南省软件工程重点实验室开放基金项目(2010KS01),云南大学研究生科研课题项目(ynuy201265,ynuy201375)资助。

朱 锐(1987—),男,博士生,CCF学生会员,主要研究领域为软件过程、形式化方法,E-mail:ray0209055@gmail.com;李 彤(1963—),男,教授,博士生导师,CCF高级会员,主要研究领域为软件过程方法与技术、软件工程,E-mail:tli@ynu.edu.cn;莫 启 男,博士生,主要研究领域为业务过程建模;张 璇 女,讲师,主要研究领域为可信过程;王一荃 女,助理研究员,主要研究领域为软件过程;林雷蕾 男,硕士生,主要研究领域为软件过程;代 飞 男,讲师,主要研究领域为软件过程,E-mail:feidai@ynu.edu.cn(通信作者)。

产品的一系列有序活动，而每个活动的属性包括相关的制品(Artifact)、资源(人或者其他资源)、组织结构和约束[7]。为了更好地管理软件过程，当前的研究者提出了以过程为中心的工程环境，利用过程进行驱动开发。同时学术界也提出大量的过程模型来对软件过程进行描述，比如李等人在文献[9]中提出一种软件演化过程元模型(Evolution Process Meta-Model，EPMM)，EPMM既可以用来建模软件演化过程，也支持定义传统的软件过程。该模型分为全局层、过程层、活动层和任务层。在过程层的建模中，该模型使用经典的条件/变迁Petri网来进行建模，通过该元模型能够很好地描述软件演化过程中的并发与迭代。通过EPMM对软件过程进行建模，在PSEE中可以有效地指导软件开发活动的展开，但是人们逐渐发现软件过程模型的实际执行和观察到的过程模型之间总是存在一定的不协调(inconsistency)，这种不协调主要有两个原因：

1)过程模型天然不完备。这种不完备主要是由于建模者无法完全获得需求，以及建模工具只能对获得信息部分建模所导致；

2)过程模型在执行时会有人的参与，由于管理问题或者人的主观性问题，都会导致过程的实施未必会按照原定的过程模型来展开。这些不协调都是无法避免的，因此，软件过程偏离的研究对于PSEE中过程的执行是重要的而且是必须的。

2 相关工作

目前国外对过程偏离(Process Deviation)的研究相对较多。但过程偏离所涉及的概念比较广，主要包括：过程感知系统(Process Aware System)、过程挖掘(Process Mining)、过程一致性检测(Process Conformance Checking)。这里的过程也不仅仅局限于软件过程，还包括业务流程，服务组装过程、物流过程等。但是，无论是软件过程，还是其他过程，其基本思想都是一致的，在非软件过程中，过程的主要作用是起到一个总体控制的流程，或者称为流程管理。而软件过程，普遍认为是最为复杂的管理过程之一，一个系统的复杂性往往可以通过与自动化程度成反比这一事实来衡量，而软件过程是自动化水平较低的过程之一。相对于研究较多的业务流过程而言，软件过程偏离的研究成果相对较少，其中具有代表性的有Cugola[10]基于一个面向工作制品(Artifact)的语言PROSYT的PSEE，其可以容忍并记录运行过程中产生的偏离，而系统使用者可以决定什么时候才用手工方式将其消除；Pohl[11]等人提出了一个集成的PSEE，其中通过在过程虚拟执行和实际执行之间引入交互协议来减少两者间的偏离；Mohammed[7]等人提出了一个基于过程的动态适应和偏离容忍的软件过程执行演化的独创方法；Sorana Cimpan等人提出了使用模糊逻辑来监控软件过程的方法。该软件过程的检测关注于给定模型的偏离检测。

国内方面，李明树在文献[6]中提到，近年来软件过程建模领域的研究人员针对上述问题作了很多有益的探索，主要的研究热点包括支持过程演化(Process Evolution)、偏离容忍(Deviation Tolerance)的PSEE、软件过程的验证和分析(主要包括过程模型的语法检查、语义正确性分析、匹配和仿真)，以及集成的软件过程模型等。杨勇和周伯生[12]为了定量控制和改进软件过程，解决由于滞后效应引起的提前量问题，提出了过程偏离阈值的计算方法。该方法以项目进度为目标控制量，增加人为修复措施，基于系统动力学方法分析了偏离控制和软件过程，建立了过程模型。顾庆和陈道蓄[13]认为软件过程是以人为中心的系统，其特点是动态性和不断演化。既定过程模型在实际执行时往往有所偏差，基于E-CSPE约束实现过程验证和偏差测量。事件约束根据过程模型定义。过程实例执行被记录为事件序列。通过分析事件序列对事件约束的覆盖和违反结果，可以计算EPD和EAD指标。EPD指标可以反映过程执行与过程模型的偏差，EAD指标则为过程演化提供依据。

纵观国内外研究现状，当前的研究往往是考虑如何对过程偏离进行处理，以及对具有偏离处理的软件过程模型进行研究，而对于过程偏离中最重要的一个部分：如何发现偏离、检测偏离，鲜有文献进行讨论和研究，这也是本论文的切入点之一。

3 软件过程偏离

随着过程模型研究的深入，人们逐渐认识到软件过程模型往往是随着开发的进行而不断与实际软件开发活动相偏离，从而逐渐失去了对实际软件开发活动的指导和规范意义[14]。近年来，软件过程建模领域的研究人员针对上述问题作了很多有益的探索，主要的研究热点包括支持过程演化、偏离容忍(Deviation Tolerance)的PSEE、软件过程的验证和分析(主要包括过程模型的语法检查、语义正确性分析、匹配和仿真)以及集成的软件过程模型等[6]。通过以上文献，我们可以知道软件过程偏离逐渐成为过程实施的研究热点。偏离是在SPM和通过PSEE所观察到的过程执行之间的一种非一致性[15]。偏离是在已定义的过程中未被描述的行为，或者是与在过程中已定义的某些约束相冲突的行为。非一致性是软件过程由于过程偏离而处于的一种状态[7]。

我们认为过程的偏离具有其必然性，如何处理偏离就是亟待解决的问题之一。偏离处理的基本思路是：一方面，从增强过程实施柔性的角度，在保证实施过程模型状态迁移正确性的前提下，对约束偏离指标和行为偏离指标低的偏离进行容忍；另一方面，从增强过程实施演化的角度，对约束偏离指标和行为偏离指标高的偏离进行处理，在过程实施演化波及效应的界定和量化支持下，使用动态过程演化算法调整实施过程模型、消除偏离。而软件过程偏离的研究重点主要集中在3个问题上：过程偏离的发现、过程偏离决策以及过程偏离处理。软件过程偏离中过程模型间的关系如图1所示。

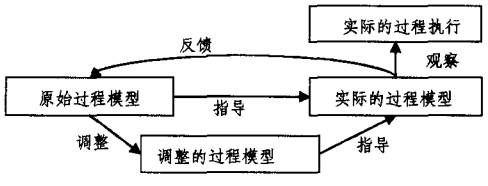

图1 软件过程偏离中过程模型间的关系

4 软件过程行为空间及其表达式

软件过程是具有行为的。文献[16]对软件行为的定义如

下：软件运行时作为主体，依靠其自身的功能对实体的施用、操作或动作称为行为。该定义说明了软件这个主体在运行时这个状态，能够根据自身所具有的功能来对外部环境进行操作或动作，我们都称为行为。而另一方面根据美国麻省 Amherst 大学计算机科学系奥斯特威尔教授（Prof. Leon Osterweil）的著名论断“软件过程也是软件”[1]，我们可知软件过程与软件一样，同样可以运行，作为主体同样可以根据其自身的功能对外部环境等客体进行操作或动作。据此，可以给出软件过程行为的定义。

定义 1 软件过程的行为是指软件过程运行时作为主体，根据其自身的活动对外部环境等主体的操作或动作。

4.1 软件过程行为建模

为了描述软件过程的行为，我们需要对过程的行为进行建模。一个软件过程具有多个行为，这些行为之间相互交织、相互作用，不仅具有层次性，还具有结构关系。一个软件过程所有的行为共同组成了其行为空间。在进程理论中，进程是软件过程与其外部环境交互的一种抽象，是软件过程的行为模型；而行为又由迁移组成，迁移的执行使得系统能够从一种状态转换到另一种状态。

定义 2（迁移关系） 迁移关系是一个三元组 $p=(p,a,p')$，可以记为 $p \xrightarrow{a} p'$，其中：

（1）$p,p' \in P$，P 为进程状态集。

（2）a 表示进程行为（动作）。

我们知道软件过程模型 $p=(N,M_0)$，其中，N 表示 Petri 网结构，M_0 表示过程模型所处的情态，如果将软件过程看作一个进程理论中的一个进程，则软件过程模型的一个情态就等价于进程的一个状态。当软件过程模型的某个活动 a 点火使得软件过程 N 从情态 M_0 转变为 M_1，我们就可以使用迁移关系表示为 $(N,M_0) \xrightarrow{a} (N,M_1)$。

定义 3（行为空间） 一个软件过程的行为空间是指该软件过程可能发生的所有行为以及行为与行为之间偏序关系的总和。我们对过程行为空间的形式化描述称为行为空间表达式。

本文由于主要考察软件过程的行为偏离，而行为偏离最为关注的是行为执行的顺序，因此提出一种基于进程代数的软件过程行为空间表达式，简称为行为空间表达式（Process Behavior Space Expression，PBSE）。现给出 PBSE 的 BNF 的语法定义：

$PBSE::=\langle activity\rangle \mid PBSE.PBSE \mid PBSE+PBSE \mid (PBSE \mid PBSE) \mid !\ PBSE \mid new\langle activity\rangle PBSE$

对于 PBSE 的语义，定义如下：我们使用 . 为顺序算子来表达顺序执行的含义，在进程的名字不会产生误解的情况下可以省略；+ 为选择算子，用以表达进程间选择执行的含义；| 为并发算子，用以表达并发执行的含义；! 为重复算子，用以表达重复执行的含义，重复次数可以为 0；new 为限制算子，用以表达将一个活动名字的辖域限制在一个行为空间表达式中；⟨activity⟩表示软件过程模型中的活动名字。另外定义特殊的活动：δ 表示过程终止活动，可以省略。因此，软件过程基本块对应的行为空间表达式如图 2 所示。

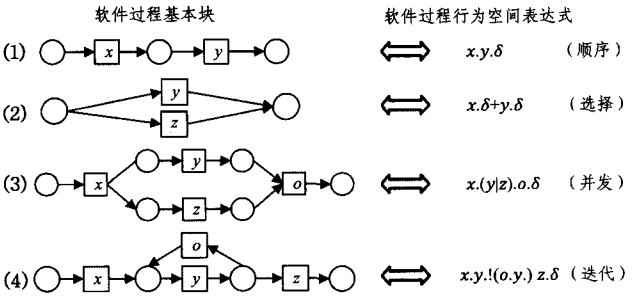

图 2 软件过程基本块与 PBSE 的对应关系

4.2 由软件过程模型生成对应 PBSE 算法

由软件过程模型生成其对应的 PBSE 算法较为复杂，主要是由于需要 Petri 网模拟执行，边执行边生成 PBSE，并且循环结构的引入也对算法产生影响，但是可以先不考虑循环结构，因此下面先给出不含循环结构的表达式生成算法，记为 noIterationPBSEProduce(N,M_b,M_e)，其中 N 为 Petri 网结构，M_b 表示开始的情态，M_e 表示结束情态（可以为 NULL）。

算法 1 生成不含迭代的 PBSE 算法

```
输入：Petri 网结构 N 及对应的情态 Mb，Me
输出：PBSE 表达式
BEGIN
  int N=getMNum(Mb，Me)/*从情态 Mb 到 Me 共有 N 种情态*/
  FOR int i=1 to N BEGIN/*从情态 Mb 到 Me 进行遍历*/
  int T=nextNextActionsNum(M[i])/*下一步可能点火的活动数量*/
    IF T==1 THEN BEGIN/*只有一个活动可以点火*/
      PBSE=PBSE.nextActions[0]/*使用.连接*/
    END
  ELSE BEGIN/*下一步 T 个活动可以点火*/
    IF beChoice == getRelation(nextActions) THEN BEGIN//若活动间为选择关系
      PBSE=PBSE.(noIterationPBSEProduce(N,M[1])+…+
            noIterationPBSEProduce(N,M[T]))//对每个分支分别进行
    END
    ELSE IF beComplicated==getRelation(nextActions) THEN
BEGIN
      PBSE=PBSE.nextActions[0]|…|nextActions[T-1]
    END
  END
END
```

不含迭代的 PBSE 生成算法如算法 1 所示，算法 2 给出含有迭代的 PBSE 算法。该算法记为 PBSEProduce(N,M_b,M_e,M_{ib},M_{ie})，其中 N 为 Petri 网结构，M_b 表示开始的情态，M_e 表示结束情态（可以为 NULL），其中 M_{ib} 表示含有循环部分的开始情态，M_{ie} 表示循环部分的结束。

算法 2 生成 PBSE 算法

```
输入：Petri 网结构 N 及对应的情态 Mb，Me，Mib，Mie
输出：PBSE 表达式
BEGIN
  PBSE=PBSE.noIterationPBSEProduce(N,Mb,Mib)//迭代前部分
  PBSE=PBSE.!(noIterationPBSEProduce(N,Mib,Mie))//迭代部分
  PBSE=PBSE.noIterationPBSEProduce(N,Mie,Me)//迭代后部分
END
```

从算法 2 可以看出，PBSE 的生成步骤其实就是将 Petri 网分为 3 个部分：开始不含迭代部分、含有迭代部分以及后边不含迭代部分，这样分解采用了分而治之的思想，其好处在于对于迭代部分只需要遍历一次就可以生成 PBSE 表达式。该算法存在一个限制条件就是只可以对含有一个迭代部分的 Petri 网进行生成，且需要在输入时将迭代部分的开始情态与结束情态作为输入。对于包含多个循环的情况，只需要将 Petri 网人为地进行分解输入，多次执行该算法即可，此处就不再进行赘述。

4.3 带状态的行为空间表达式

根据上述算法分析可以得出从 Petri 网如何生成 PBSE，但是考虑到一个软件过程是可以执行的，而上述定义的 PBSE 表达式目前还没办法表达软件过程每一步执行的信息，因此，考虑在 PBSE 中加入已经发生的活动记录信息，来对软件过程的每一步进行描述。

我们以图 2 中第 4 个软件过程基本块为例，来介绍添加状态信息后的 PBSE 是如何描述软件过程执行以后的行为空间的。其中使用 $\sharp$ 来表达未执行任何记录。

$M_0 \Leftrightarrow E_{[\sharp]} = x.\ y.\ !\ (o.\ y.)z.\ \delta$

$M_1 \Leftrightarrow E_{[\sharp y]} = y.\ !\ (o.\ y.)z.\ \delta$

$M_2 \Leftrightarrow E_{[\sharp xy]} = !\ (o.\ y.)z.\ \delta$

$M_3 \Leftrightarrow E_{[\sharp xyo]} = y.\ !\ (o.\ y.)z.\ \delta$

$M_4 \Leftrightarrow E_{[\sharp xyoy]} = !\ (o.\ y.)z.\ \delta$

$M_5 \Leftrightarrow E_{[\sharp xyoyz]} = \delta$

其中，M_0 到 M_5 分别对应的软件过程执行情态如图 3 所示。

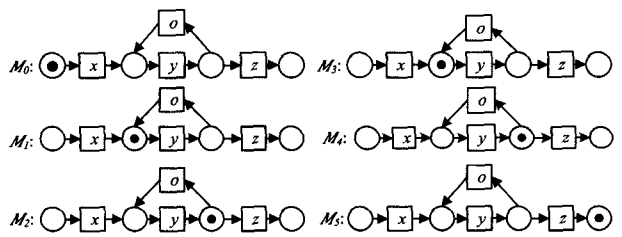

图 3 M_0 到 M_5 分别对应的软件过程执行情态

通过上图可知，PBSE 是可以用来描述动态执行的软件过程的每一步的行为空间的。为了描述软件过程执行的每一步的行为空间，PBSE 必须能够进行迁移操作，这种迁移主要是通过迁移规则来完成的。因此下边给出 PBSE 的迁移规则。

定义 4(迁移规则)

(1)设 $PBSE_{[\sharp]} = \pi.\ PBSE'$，且 $\exists PBSE \xrightarrow{\pi} PBSE'$，则有 $PBSE_{[\sharp\pi]} = PBSE'$。

(2)设 $PBSE_{[\sharp]} = (\pi + \tau).\ PBSE'$，且 $\exists PBSE \xrightarrow{\pi} PBSE'$，则有 $PBSE_{[\sharp\pi]} = PBSE'$。

(3)设 $PBSE_{[\sharp]} = (\pi \mid \tau)\ PBSE'$，且 $\exists PBSE \xrightarrow{\pi,\tau \wedge \pi \gg \tau} PBSE'$，则有 $PBSE_{[\sharp\pi\tau]} = PBSE'$。其中 $\gg$ 表示活动发生先于。

(4)设 $PBSE_{[\sharp]} = !\ (\pi.\ \tau.)\varphi.\ PBSE'$，且 $\exists PBSE \xrightarrow{\pi} PBSE'$，则有 $PBSE_{[\sharp\pi]} = \tau.\ !(\pi.\ \tau.)\varphi.\ PBSE'$。

(5)设 $PBSE_{[\sharp]} = !\ (\pi.\ \tau.)\varphi.\ PBSE'$，且 $\exists PBSE \xrightarrow{\varphi} PBSE'$，则有 $PBSE_{[\sharp\varphi]} = PBSE'$。

5 基于行为空间表达式的偏离发现

首先给出一条关于软件过程偏离的公理。

公理 1 对于未发生偏离的软件过程而言，实施过程模型一定能够模拟观察过程模型的行为。

该公理是说对于软件过程而言，如果它没有发生偏离，其实施过程模型的行为一定比观察过程模型丰富，这种丰富主要体现在观察过程模型 OPM 的执行情况只能是实施过程模型 EPM 的一个实例。也就是说对于未偏离的观察过程模型的所有行为都应该在实施过程模型中已经定义好。

定义 5(实验关系[17]) 关系 $\Rightarrow$ 和 $\overset{s}{\Rightarrow}$，$s \in Act^*$ 定义如下：

(1)$P \Rightarrow Q$ 表示存在长度大于等于零的迁移序列 $P \rightarrow \cdots \rightarrow Q$，或更形式地，$\Rightarrow \overset{def}{=} \rightarrow^*$，其中 $\rightarrow^*$ 为 $\rightarrow$ 的自反闭包。

(2)令 $s = a_1 \cdots a_n$，则 $P \overset{s}{\Rightarrow} Q$ 表示 $P \Rightarrow \xrightarrow{a_1} P_1 \cdots \Rightarrow \xrightarrow{a_n} P_n \Rightarrow Q$，或更形式地，$\overset{s}{\Rightarrow} \overset{def}{=} \Rightarrow \xrightarrow{a_1} \Rightarrow \cdots \xrightarrow{a_n} \Rightarrow$。

定义 6(弱模拟[17]) 设 $\boldsymbol{S}$ 是进程空间 ρ 的二元关系。如果对于任意的 $P\boldsymbol{S}Q$ 下面的条件成立，则称 $\boldsymbol{S}$ 是一个弱模拟。

如果 $P \overset{e}{\Rightarrow} P'$，那么存在 $Q' \in \rho$ 满足 $Q \overset{e}{\Rightarrow} Q'$ 并且 $P'\boldsymbol{S}Q'$。

当存在这样的弱模拟关系 $\boldsymbol{S}$ 满足 $P\boldsymbol{S}Q$ 时，我们说 Q 弱模拟 P。

根据定义 5 与定义 6，我们可以给出软件过程未偏离的定义。

定义 7(软件过程未偏离) OPM 未发生偏离 iff OPM $\boldsymbol{S}$ EPM。

定义 7 说明当 OPM $\boldsymbol{S}$ EPM 时，即 EPM 弱模拟 OPM 时，OPM 所有的行为都能够在 EPM 中出现。通过软件过程行为空间表达式以及弱模拟的方法能够快速判断过程偏离，甚至不需要去构建观察过程模型就可以进行判定。判定算法如下。

算法 3 偏离判定检测算法

```
BEGIN
  E :=producePBSE(EPM)
  FOR i :=1 TO n DO BEGIN / * 遍历每一条轨迹 * /
    FOR j :=1 TO σi. Length DO BEGIN
      IF (isMeetTransition(σi. firstAction)) THEN BEGIN
        transition(σi. firstAction)
      END ELSE BEGIN
        takePlaceDeviation()/ * 发生偏离 * /
      END
    END
  END
END
```

6 偏离诊断处理策略

本节主要讨论过程偏离的处理，对于过程偏离的处理有 3 个问题需要解决：问题 1：需要纠偏的标准；问题 2：偏离处理的准则；问题 3：对不同情况发生的偏离应如何解决。第 1

个问题对于纠偏标准，就是说一个过程发生了偏离应判断是否需要对其处理；第 2 个问题偏离处理的准则的意思是如果需要进行处理应以什么为处理准则；而第 3 个问题是指确定准则后对不同的情况我们应该如何进行解决。

对于问题 1，我们知道有些偏离是不需要进行纠偏的，而有些则需要，过程的偏离不一定就是错的。判断是否需要纠偏就涉及到一个需要纠偏的标准，通过该标准能够对过程偏离是否需要纠偏来进行判断。本文提出了一种基于目标的过程行为纠偏准则，该准则首先对过程目标进行定义，然后通过过程执行对过程执行的产品进行形式化描述，并且通过与目标描述进行对比来判断是否需要纠正偏离。

定义 8(目标)　目标 T 为一个四元组 $T=(I,D,S,A)$，其中 I 表示目标的唯一标识；D 表示目标 T 的目标描述集，每个描述使用一阶谓词进行定义；S 表示目标 T 的状态，包括完成还是未完成，即 True 还是 False；A 表示该目标与哪些活动具有对应关系。

定义 9(目标集)　目标集 G 是所有目标的集合。

定义 10(过程目标集)　设软件过程 $P=(C,A,F;M)$，则其对应的过程目标集 $PG=\{T\mid \forall_{T\in G\wedge p\in P}T.\,a\in p.\,A\}$，过程目标集是指过程 P 所对应的所有目标的集合。

在文献[18]中，Jacques 对软件过程进行了定义，该定义如下：

通过软件过程的定义我们知道，软件过程是由一系列活动组成，并且这些活动将输入转化为一定的输出，我们将这些输出称为“产品”。因此可以给出过程输出的定义。

定义 11(过程输出)　一个软件过程的输出是一系列产品的集合。对这些产品的语言描述，我们称之为过程输出描述 $POD=\{o\mid$过程 P 的所有产品描述$\}$，其中 P 指软件过程。

通过以上定义就可以给出判定需要纠偏的唯一标准：过程目标集与过程输出是否协调。如果过程目标集与过程输出是协调的，我们认为即使存在偏离，也是可以容忍的；但是如果过程目标与过程输出是非协调的，我们认为这样的过程必须要进行偏离处理。我们将需要纠偏定义为 ND，则需要纠偏的判定标准的形式化描述如下：

$$\frac{\forall t\in PG\wedge(\exists o\in POD\rightarrow d\in t.\,D)}{ND}$$

其中，PG 表示过程目标集，→表示匹配。

对于问题 2，偏离处理总体来说有两个标准：

a. 以目标为准。当发生偏离的时候，人们经过讨论，觉得当初定的目标是正确的，应该按照该目标执行。

b. 以现实为准。当发生偏离的时候，人们经过讨论，觉得当初定的目标存在问题，需要对目标进行调整。

对于问题 2 的标准其实是当过程偏离发生时过程支撑系统如何决策的标准，这些决策标准有很多，例如文献[19]中介绍了 5 种偏离决策的方法：中止(Abort)、通知用户并且中止、询问用户、通知用户并且继续、继续。但是由于我们所讨论的均在 PSEE 这个基调下，因此在偏离时应首先以过程为准，即应该采用 a 准则，即 a 准则的权重应大于 b。

对于问题 3 一般偏离处理方式有 3 种：

1. 容忍(tolerating)：在不违背目标的情况下继续。($\Im$)

2. 调整过程模型以适应下次过程模型执行：过程建模者在过程执行后发现过程输出与过程目标集不能匹配，且过程建模者认为过程模型存在问题，因此需要对过程模型进行调整。($\Re$)

3. 对本次过程执行进行补充与完善：过程建模者在过程执行后发现过程输出与过程目标集不能匹配，且过程建模者认为过程执行并未完成目标需要加入某些活动来完成此目标。($\aleph$)

上述 3 种处理方式后边字母表示该方式的简记。处理方式 1 $\Im$就是通过过程建模者对过程输出与目标集进行匹配发现，过程目标集中的目标在过程输出中都可以体现，这样过程支撑系统就采用容忍的处理方式。处理方式 2$\Re$是过程模型存在问题，需要对过程模型进行改进，这个改进过程需要过程建模者进行完善。同理，处理方式 3 $\aleph$代表过程实例化存在问题，这个过程同样需要人的参与，需要对过程实例进行补充与修改。综上，3 种方式是相互补充、相互完善的，3 种处理方式都需要人的参与，都需要人为地判断过程目标集与过程输出是否匹配才能完成过程偏离的整个判断过程。

7　案例研究

本节针对本文提到的分析方法以及偏离处理问题，给出相关的案例研究，通过该案例的研究与分析可以对整篇论文的思想以及解决方案进行进一步的了解。本文主要将进程代数思想应用于过程偏离的研究中，提出了过程行为空间表达式来构建软件过程的行为空间，并且通过过程空间表达式可以有效地发现偏离，最后对发现的偏离进行处理。

首先已知以下条件：

1. 实施过程模型 M_1 如图 4 所示，该图为某公司需求分析过程模型，其中字母代表的含义如图中所示；

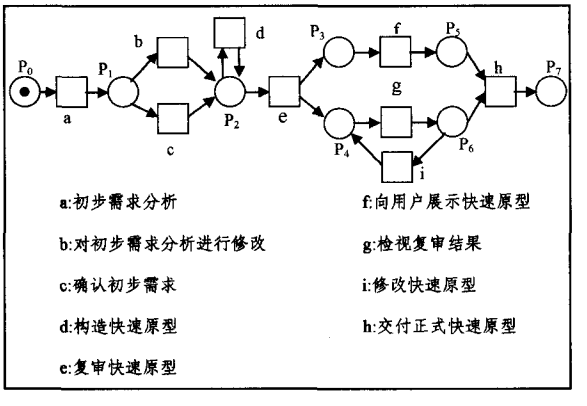

图 4　实施过程模型 M_1

2. $PG(M_1)=\{t_1,t_2,t_3,t_4,t_5,t_6,t_7,t_8,t_9\}$，其中：$t_1=(1,\{d_1,d_2\},F,a)$，$t_2=(2,\{d_3,d_4,d_5\},F,b)$，$t_3=(3,\{d_3,d_4,d_5\},F,c)$，$t_4=(4,\{d_6,d_7,d_8,d_9\},F,d)$，$t_5=(5,\{d_3,d_{10},d_{11}\},F,e)$，$t_6=(6,\{d_{12}\},F,f)$，$t_7=(1,\{d_{13},d_{14}\},F,g)$，$t_8=(8,\{d_{15},d_{16},d_{17}\},F,h)$，$t_9=(9,\{d_{18}\},F,i)$。

3. 目标集：$POG(M_1)=\{o_1,o_2,o_3,o_4,o_5,o_6,o_7,o_8,o_9,o_{10},o_{11},o_{12},o_{13},o_{14},o_{15},o_{16}\}$。

以及目标匹配规则：

$o_1\rightarrow d_1,d_2$；$o_4\rightarrow d_3$；$o_5\rightarrow d_4,d_6$；$o_6\rightarrow d_5,d_7$；$o_7\rightarrow d_8$；

$o_8 \rightarrow d_9$；$o_9 \rightarrow d_{10}$；$o_{10} \rightarrow d_{11}$；$o_{11} \rightarrow d_{12}, d_{13}, d_{14}$；$o_{15} \rightarrow d_{17}, d_{18}$

已知条件有很多，主要原因是软件过程是一个复杂的过程，会涉及到多个方面的信息。

步骤一　根据算法2可以得到EPM对应的PBSE：$P_{[\#]}=a.(b+c).(!\ d).e.(f|g!(i.g.)).h.\delta$

对 $\sigma_1=abesgh$，有：

$E_{[\#]}=a.(b+c).(!d).e.(f|g!(i.g.)).h.\delta$

$E_{[\#a]}=(b+c).(!d).e.(f|g!(i.g.)).h.\delta$

$E_{[\#ab]}=(!d).e.(f|g!(i.g.)).h.\delta$

$E_{[\#abe]}=(f|g!(i.g.)).h.\delta$

$E_{[\#abesg]}=!(i.g.).h.\delta$

$E_{[\#abesgh]}=\delta$

通过以上对轨迹 σ_1 在PBSE中进行规约，可知，s 无法进行规约，发生偏离。且 s 之后的部分能够正常规约，那么 s 发生了替换偏离。

对 σ_2、σ_3、σ_4 有同样结果，这里不再赘述。且由于日志 W 为完全的，那么可知活动 c 从未执行过，因此活动 c 被缺失，发生了缺失偏离。

步骤二　发生偏离后，就需要对偏离进行处理。由已知条件2以及条件3可知，活动 c 与活动 b 处于选择结构中，二者的目标是完全一致的，因此活动 c 的缺失未影响到过程输出描述POD与过程目标集PG的匹配，因此活动 c 的缺失不需要进行处理。但是活动 f 发生了替换偏离。由于活动 f 对应的目标描述为 d_{12}，f 的缺失就导致过程输出 $o_{11} \rightarrow d_{12}$，$d_{13}$，$d_{14}$ 无法匹配，因此我们根据判定需要纠偏的唯一标准确认此时是否需要纠偏。根据偏离处理的标准可以分为两种情况考虑：

情况一　假设采用偏离处理标准a，则可知当初定的目标是正确的，应该按照该目标执行。

目标正确，即原始的EPM是没有问题的，因此处理方式将采用$\aleph$，即需要对OPM进行补充和调整。偏离实质上是因为活动 s 替换了活动 f 所导致的，因此需要对 s 进行补充，可知活动 s 所对应的目标为 $t_6'=(6,\{d_{12}\},F,s)$。这样就符合我们预先定义的过程目标集。

情况二　假设采用偏离处理标准b，则可知当初定的目标存在问题，需要对目标进行调整。

对于情况二，即过程目标集PG存在问题，我们将采用偏离处理方式$\Re$进行处理，因为目标 t_6 中的描述 d_{12} 没有输出可以匹配，所以我们只需要将EPM中的描述 d_{12} 删除即可。而活动 s 替换 f 后无论任何输出，其目标均可以被匹配，这样在下一次过程模型的实例化时就不会产生过程目标集与过程输出不匹配的情况，即使存在替换偏离，我们也可以采用处理方式$\Im$进行处理。

综上，整个过程偏离从发现到检测再到最后进行处理的方法叙述完毕。

结束语　由于当前软件过程领域对偏离研究仍然存在一定的不足，现有的文献并不能有效地解决软件过程偏离问题，因此本文以软件演化过程元模型(EPMM)为基础，借鉴进程代数的弱互模拟思想，提出一种软件过程偏离的诊断方法，以解决以下问题：

1. 在软件过程偏离发现方面，提出过程行为空间表达式来对软件过程的行为空间进行构造。为了从行为视角对OPM和EPM进行行为比较，本文采用进程代数中的互模拟关系作为行为比较的标准，提出过程行为空间表达式来对过程行为进行描述以及基于进程代数中互模拟思想的过程偏离的诊断方法。

2. 在软件过程偏离处理方面，综合当前的过程偏离研究的相关文献，提出了过程偏离类型的划分。基于偏离类型划分提出了偏离处理策略，通过偏离处理策略来对过程偏离进行处理。

通过上述分析，本文仍然存在一定的不足，下一步将对其进行改善，主要包括：

1. 根据本文提出的理论来构建形式化的验证工具，对实际的问题进行解决。

2. 对本文提出的算法，在验证工具中进行具体的实现。

3. 从数学角度而言，需要对本文涉及到的某些思想进行数学证明，比如软件过程行为空间表达式与Petri网的行为空间之间的一致性问题。应从语义层进行相应的证明。

4. 从偏离的处理而言，当前提出的偏离处理完全是需要人来进行参与，是否可以给出一个基于规则库的偏离处理策略也是有待研究的方面。

参 考 文 献

[1] Osterweil L J. Software Processes are Software too[C]// Proceedings of the 9th International Conference on Software Engineering. ACM Press, New York, NY, 1987: 2-13

[2] Montangero C, Derniame J C, Kaba B A, et al. The software process: Modeling and technology[C]// Derniame J C, BAK, Wastell D G, eds. Proc. of the software Process: Principles, Methodology, and Technology. Springer-Verlag, 1999: 1-14

[3] SEI. CMMI for Development, Version 1.2-Improving Process for Better Products[S]. SEI, CMU, 2006

[4] Humphrey W S. A Discipline for Software Engineering[M]. Boston: Addison-Wesley Longman Publishing Co., Inc., 1995

[5] Arbaoui S, Derniame J-C, Fav O, et al. A comparative review of process-centered software engineering environments[J]. Annal of Software Engineering, 2002, 14(1-4): 311-340

[6] 李明树，杨秋松，翟健. 软件过程建模方法研究[J]. 软件学报，2009，20(3)：524-545

[7] Kabbaj M, Lbath R, Coulette B. A deviation-tolerant approach to software process evolution[C]// Proceedings of the 9th international workshop on Principles of software evolution (IWPSE 2007). ACM Press, 2007: 75-78

[8] Lonchamp J. A structured conceptual and terminological framework for software process engineering[C]// Proc. of the ICSP. 1993. 4153

[9] Tong Li. An Approach to Modeling Software Evolution Processes [M]. Spring-Verlag, Berlin, 2008

[10] Cugola G. Tolerating deviations in process support system via flexible enactment of process models[J]. IEEE Trans. on Software Engineering, 1998, 24(11): 982-1001

[11] Pohl K, Weidenhaupt K, Domges R, et al. PRIME—Toward

process-integrated modeling environments[J]. ACM Trans. on Software Engineering and Methodology, 1999, 8(4): 343-410

[12] 杨勇，周伯生. 基于系统动力学的软件过程偏离控制[J]. 计算机工程与设计，2011，32(5)：1684-1690

[13] 顾庆，陈道蓄. 基于事件约束的软件过程验证[J]. 软件学报，2005，16(10)：1735-1742

[14] Dowson M, Fernström C. Towards requirements for enactment mechanisms[M]. Software process technology. Springer Berlin Heidelberg, 1994: 90-106

[15] da Silva M A A, Bendraou R, Robin J, et al. Flexible Deviation Handling during Software Process Enactment[C]// Fifteenth IEEE International EDOL Conference. IEEE, 2011: 34-41

[16] 屈延文. 软件行为学[M]. 北京：电子工业出版社，2004

[17] 米尔纳，林惠民. 通信与移动系统：π 演算[M]. 北京：清华大学出版社，2009

[18] Lonchamp J. A structured conceptual and terminological framework for software process engineering[C]// Proceedings of the Second International Conference on the Software Process, 1993, 2: 41-53

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

（上接第 35 页）

通过以上实例，我们可以看到 Smart SEP 平台给教师与学生之间提供了一个类似“电子黑板”的功能，通过展示权的管理实现了教师与学生之间同步的、动态的、互动的图形建模教学过程。因此，本文所提出的方法是可行、有效的。

表 2 显示了在线同步教学平台 Smart SEP 与其他几种传统的教学解决方案特点的对比。

表 2 几种解决方案对比

解决方案	同步性	动态性	交互性
文档、讲义	没有	没有	没有
视频、远程桌面	有	有	没有
协同工作空间	有	没有	有
Smart SEP 平台	有	有	有

结束语 本文介绍了一种基于 Web 图形操作记录与回放的在线同步教学方法，讨论了高保真的图形操作记录、自适应的图形操作回放、正常参与者和迟到者如何进行同步控制等问题；同时，本文通过在线同步教学平台 Smart SEP 的实例展示了师生之间如何进行图形化的教学互动，并论证了本文方法的可行性和有效性。

未来的工作，我们将从以下两方面开展。第一，支持更多类型的 Web 图形技术，从而扩大本文方法的应用范围，如 HTML5 的 Canvas 等新绘图技术。第二，完善同步机制，使之更符合实际的教学过程，例如增加本地学习区，使得学生能够把协同学习区的内容复制下来并进行本地修改尝试，而不影响协同学习区中正常的教学互动过程。

参 考 文 献

[1] Russell D M, Klemmer S. Will massive online open courses (moocs) change education? [C]// Extended Abstracts on Human Factors in Computing Systems. Paris, France, 2013: 2395-2398

[2] Wang Zhou-xiu, Hu Ting, Xu Ya-feng, et al. A designing and research of future classroom learning support system based on cloud computing technology[C]// Proceedings of the 2013 Third International Conference on Intelligent System Design and Engineering Applications. Hongkong, China, 2013: 50-53

[3] edX [EB/OL]. https://www.edx.org/

[4] Coursera [EB/OL]. https://www.coursera.org/

[5] Udacity [EB/OL]. https://www.udacity.com/

[6] Notemate [EB/OL]. http://www.tiaozhanbei.net/project/18427/

[7] Willey K, Gardner A. Collaborative Learning Frameworks to Promote a Positive Learning Culture[C]// Proceedings of the 2012 IEEE Frontiers in Education Conference. Washington D. C., USA, 2012: 1-6

[8] Zhang Ying, Huang Gang, Zhang Nu-yun, et al. SmartTutor: Creating IDE-based interactive tutorials via editable replay[C]// Proceedings of the 31st International Conference on Software Engineering. Vancouver, Canada, 2009: 559-562

[9] 陈德健，孙艳春，黄罡，等. 基于操作记录与回放技术的远程同步教学工具[J]. 计算机工程与应用，2013，49(6)：65-71

[10] Leshed G, Haber E M, Matthews T, et al. CoScripter: automating & sharing how-to knowledge in the enterprise[C]// Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. Florence, Italy, 2008: 1719-1728

[11] Little G, Lau T A, Cypher A, et al. Koala: capture, share, automate, personalize business processes on the Web[C]// Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. California, USA, 2007: 943-946

[12] iMacros [EB/OL]. http://wiki.imacros.net/Main_Page/

[13] Khalili A, Auer S, Hladky D. The RDFa Content Editor-From WYSIWYG to WYSIWYM[C]// Proceedings of the 2012 IEEE 36th Annual Computer Software and Applications Conference. Izmir, Turkey, 2012: 531-540

[14] Atif M, Mousavi M R. Formal Specification and Analysis of Accelerated Heartbeat Protocols[C]// Proceedings of the 2012 Summer Computer Simulation Conference. Ottawa, Canada, 2012: 403-412

[15] Gutwin C, Graham T C N, Wolfe C, et al. Gone but not forgotten: designing for disconnection in synchronous groupware[C]// Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work. Georgia, USA, 2010: 179-188

[16] Kim Y. Web Information Extraction by HTML Tree Edit Distance Matching[C]// Proceedings of the 2007 International Conference on Convergence Information Technology. Gyeongju, Korea, 2007: 2455-2460

[17] SEForge [EB/OL]. http://www.seforge.org/