软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2018,29(11):3455−3483 [doi: 10.13328/j.cnki.jos.005304]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

数据驱动的双层次软件过程挖掘方法*

朱 锐[1,2], 李 彤[1,2], 莫 启[1], 何臻力[1], 于 倩[1], 王一荃[1]

[1](云南大学 软件学院,云南 昆明 650091)

[2](云南省软件工程重点实验室(云南大学),云南 昆明 650091)

通讯作者: 李彤, E-mail: tli@ynu.edu.cn

摘 要: 为了解决软件过程数据因活动信息及案例属性的缺失而无法应用传统过程挖掘方法的问题,以软件过程数据为研究对象,提出了一种双层次的软件过程挖掘方法.在活动层,提出加权结构连接向量模型对过程日志进行向量化,通过平均活动熵来确定过程日志模糊聚类的结果,将聚类结果作为活动信息支持后续挖掘工作的开展;在过程层,以启发式关系度量为基础,针对非完全循环进行研究,提出了过程层单触发序列循环划分的日志完备性条件,并进一步给出了循环归属的度量方法.基于大量真实软件过程数据的实验结果表明了双层次的软件过程挖掘方法的可行性及正确性.

关键词: 软件过程数据;软件过程挖掘;模糊聚类;启发式关系度量

中图法分类号: TP311

中文引用格式: 朱锐,李彤,莫启,何臻力,于倩,王一荃.数据驱动的双层次软件过程挖掘方法.软件学报,2018,29(11):3455−3483. http://www.jos.org.cn/1000-9825/5304.htm

英文引用格式: Zhu R, Li T, Mo Q, He ZL, Yu Q, Wang YQ. Data-Driven bilayer software process mining. Ruan Jian Xue Bao/ Journal of Software, 2018,29(11):3455−3483 (in Chinese). http://www.jos.org.cn/1000-9825/5304.htm

Data-Driven Bilayer Software Process Mining

ZHU Rui[1,2], LI Tong[1,2], MO Qi[1], HE Zhen-Li[1], YU Qian[1], WANG Yi-Quan[1]

[1](School of Software, Yunnan University, Kunming 650091, China)

[2](Key Laboratory in Software Engineering of Yunnan Province (Yunnan University), Kunming 650091, China)

Abstract: To address the issue of difficulty in applying the traditional process mining on software process data due to the deficiency of activity and case attribute, this paper focus on the software process data and proposes a bilayer software process mining approach. In the mining activity layer, a weighted structured linked vector mode is proposed to vectorize the process log. The result of fuzzy clustering, which can be regarded as activity information, is determined by the average activity entropy. In the process layer, based on the heuristic relation metrics, this paper studies the non-complete cycle situation and presents the single firing sequence of loop dividing condition of log completeness, and then proposes a method to measure the affiliation of loop. The real-world data sets are used to show the effectiveness and correctness of the proposed bilayer software process mining.

Key words: software process data; software process mining; fuzzy clustering; heuristic relation metrics

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[1,2].软件过程是指用于开发和维护软件产

* 基金项目: 国家自然科学基金(61662085, 61862065); 云南省教育厅科学研究基金(2017ZZX227); 云南大学数据驱动的软件工程省科技创新团队项目(2017HC012); 阿里巴巴青年学者支持计划

Foundation item: National Natural Science Foundation of China (61662085, 61862065); Project of Yunnan Provincial Department of Education Science Research Fund (2017ZZX227); Yunnan University Data-Driven Software Engineering Provincial Science and Technology Innovation Team Project (2017HC012); Alibaba Young Scholars Support Program

收稿时间: 2017-01-06; 修改时间: 2017-04-14; 采用时间: 2017-05-07; jos 在线出版时间: 2018-06-07

CNKI 网络优先出版: 2018-06-07 14:53:30, http://kns.cnki.net/kcms/detail/11.2560.TP.20180607.1453.001.html

© 中国科学院软件研究所 http://www.jos.org.cn

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[3].传统软件过程研究主要分为两类:一类是以能力成熟度集成模型 CMMI(capability maturity model integration)[1]为代表的软件过程评估和改进模型;另一类是软件过程建模,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[4].传统软件过程模型的建立主要通过宏观上自顶向下的人为建模进行完成,从建模思想到模型,再从模型到过程实施,最后生成数据.

随着对软件过程研究的深入,人们逐渐发现,虽然“过程驱动开发”是一种非常好的开发范式[5],但是传统人为建模的方法却存在许多问题.这些问题主要集中在:软件过程模型的设计工作极其复杂、易于出错、模型生命周期短暂、人们对于实际过程和过程模型间的差异不敏感、对过程工程师的要求较高;同时,软件过程还在动态地演化中[6−8].随着当前软件系统越来越复杂,一个优质过程模型的建立越来越像“一种艺术而非科学”[9].因此,传统的软件过程建模方法已经不能有效满足当前大数据时代软件工程领域对“过程建模”的需求.如何能够自动化地从软件开发组织已有的过程数据中挖掘出过程模型,已经成为当前软件过程研究的热点.

软件过程数据作为挖掘的基础,是伴随着软件开发、演化、维护以及测试等活动所产生的[10,11],它是软件工程数据的子集.文献[12,13]认为,软件工程数据包括可行性分析与需求分析文档、设计文档、使用说明、软件代码和注释、软件版本及其演化数据、测试用例和测试结果、软件开发者之间的通信、用户反馈等.

区别于软件工程数据,我们将软件工程数据中存储软件过程行为信息的数据称为软件过程数据,比如谁创建、访问或者改变这些文档,任务具体在什么时间提交并完成等[7].维护软件过程数据的管理系统包括软件配置管理系统(software configuration management system,简称 SCM)、项目管理系统(project management software,简称 PMS)、缺陷跟踪系统(defect tracking system,简称 DTS)等.这些包含事件执行语义的过程数据为软件过程挖掘奠定了基础.

软件过程挖掘(software process mining)是指在软件过程数据的驱动下,能够自动地发现软件过程模型,进而帮助软件工程师更好地识别、理解、分析、优化实际执行的软件过程,最终达到软件过程改进并提升软件产品质量的目的.本文重点关注如何从软件过程数据中高效地挖掘出过程模型以及如何对挖掘结果加以验证.与传统业务过程领域过程挖掘相比,软件过程挖掘具有以下特征.

1) 数据基础不同,软件过程数据与业务过程数据的来源、组成以及属性方面存在差异;
2) 挖掘侧重的方面不同,软件过程实例相对较少,并且具有大量的并发、迭代以及复杂结构;
3) 验证标准存在一定差异,软件过程挖掘要求不仅能够获取模型,同时更加重视获取模型的速率、简洁度和精确度等质量属性;
4) 挖掘目的不同,软件过程以人为中心,最终目的是为了提升软件产品质量.

自动化地发现软件过程模型的方法,已成为软件组织挖掘潜在过程模型、分析过程缺陷、改善产品质量的重要使能技术(enabling technology).当前已有的软件过程挖掘研究主要借鉴业务过程领域的挖掘方法,未能对软件过程数据进行针对性的分析,缺少统一框架,因此无法建立完整的软件过程挖掘理论基础,使得挖掘结果不能达到预期标准[9].为此,本文针对软件过程挖掘的特点及问题,提出了一种双层次的软件过程挖掘方法.本文的贡献可以分为以下几个方面.

1) 提出了双层次的软件过程挖掘方法.相对于现有的上下文无关的过程挖掘方法或者仅能挖掘出粗粒度的增量式软件过程挖掘方法,在支持缺失任务的活动信息及案例信息的过程挖掘的同时,又保证了挖掘结果的正确性和精确度.
2) 在活动层,为了能够从过程日志中发现活动信息,提出了基于聚类的活动发现方法.该方法对过程日志中每个事件中的特征词进行抽取,一个过程日志被转化为特征词集;为了区分特征词的重要程度,提出加权结构连接向量模型(weighted structured linked vector model,简称 WSLVM)为每个特征词赋予相应的权值;提出平均活动熵来衡量模糊聚类结果,并将聚类结果作为活动与事件进行关联.
3) 在过程层,提出了非完全循环情况下的事件日志的完备性、循环归属等判定方法,完善和补充了启发

© 中国科学院软件研究所 http://www.jos.org.cn

式的单触发序列的循环实例划分.

本文第 1 节重点阐述软件过程挖掘的基本概念以及传统软件过程建模和现有过程挖掘方法间的关系.第 2 节对软件过程挖掘中涉及到的基本概念——软件过程模型、过程数据以及过程日志进行表述.第 3 节详细论述双层次软件过程挖掘的方法框架,该框架将软件过程挖掘分为活动层和过程层.第 4 节针对活动层挖掘展开研究,重点关注如何将过程日志中的活动信息进行抽取.第 5 节针对启发式过程层挖掘中遇到的非完全循环情况及划分日志完备性进行讨论.第 6 节就本文提出的方法,利用真实的软件过程数据进行针对性的实验及分析.第 7 节讨论相关工作并与本文方法进行比较.第 8 节对全文进行总结,并展望未来的工作.

1 软件过程挖掘

软件过程挖掘是软件过程与过程挖掘的一个交叉研究领域.软件过程挖掘在宏观上是自底向上的,先有数据,后有模型;通过数据来发现过程模型.当前现有大量的软件过程模型与元模型[4,14,15]使得对于模型的表达和描述已经不再成为问题,当前已有的软件过程执行日志和具有大数据特征的软件过程数据[16,17]为数据的分析和挖掘奠定了基础,当前,困难的软件过程模型构建为自动化地挖掘软件过程模型提出了强烈的诉求.综合上述原因,软件过程数据的分析与挖掘应该给予大量的重视.

过程挖掘理论与方法是软件过程挖掘的基础.过程挖掘的理念是指通过从事件日志中提取出知识,从而去发现、监控和改进实际过程(即非假定过程)[9].如图 1 所示,过程挖掘建立了两种连接:一是实际过程与其数据的;二是实际过程与过程模型的连接.过程挖掘研究有 3 个应用场景:过程发现、符合性检查和过程改进[18].过程发现是指根据一个事件日志生成一个模型,并不使用任何先验信息;符合性检查是指将一个已知的过程模型与这个模型的事件日志进行对比,进而被用来检查事件在日志中的实际情况是否和模型相吻合;过程改进是指使用一些事件日志中记录的实际过程来扩展或者改进现存的过程模型.通过过程挖掘,能够有效地分析过程中的瓶颈、发现隐藏的非一致性、检测依从性(compliance)、解释偏离、预测行为以及引导用户朝着“更好的”过程前进[19].其中,过程模型发现算法是过程挖掘的核心内容.当前,人们提出了大量过程模型发现算法,如α算法[20]、启发式挖掘[21]、基于区域的挖掘[22,23]、遗传过程挖掘[24]、模糊挖掘[25]、ILP 挖掘[26]等.

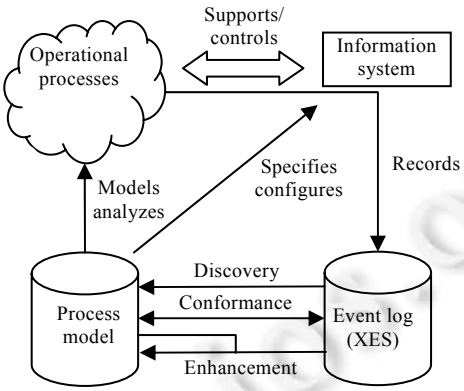


Fig.1 Process mining overview[9,24]

图 1 过程挖掘概览[9,24]

一般认为,软件过程挖掘起源于 90 年代后期,Cook 和 Wolf 最早关注了软件过程模型的发现问题,但由于当时技术和数据的限制,使得无法从中正确有效地发现软件过程模型.该文也被认为是第 1 篇正式地、较为系统地介绍过程挖掘方法的文章,并提出了相应的解决方案.但是这些方法没有考虑并发结构,故只限于顺序过程的发现[27].文献[28]中,Rubin 等人将软件过程挖掘分为“软件过程”挖掘和“软件”过程挖掘两种模式.其中,“软件”过程挖掘是指对“软件”进行过程挖掘,也就是对于软件进行过程挖掘时所挖掘出的过程是固化在软件系统中

© 中国科学院软件研究所 http://www.jos.org.cn

的业务流程进行挖掘;而“软件过程”挖掘是指基于过程数据自动化地发现软件过程模型的方法,这是本文研究的重点,简称为软件过程挖掘.当前已经存在一定的文献对软件过程挖掘的相关问题进行讨论,在文献[29]中,Garg和Bhansali采用基于解释的学习方法(explanation-based learning),将具体的软件实施过程泛化,从软件开发过程的历史数据发现潜在的过程模型.文献[10]中,Hindle 等人在 Unified Process 视图展示软件的开发过程.文献[30]对自动地发现软件过程模型的方法进行了总结,认为主要包括语法推理、过程挖掘方法、参考模型的方法,并提出一个软件过程挖掘框架.文献[31]提出一种利用软件过程评估数据来挖掘软件过程评估的过程框架.另一种较流行的软件过程挖掘的方法就由 Kindler 提出的一种增量挖掘法[7,8,32],该方法可以通过一些过程数据发现一个粒度较粗的过程模型,并且一旦有其他数据产生,就可以重新细化原来的过程模型.这样,随着日志数量的增加,过程模型的精细程度也就越来越高.Kindler 已经将该方法应用于软件过程挖掘,并且通过挖掘开源软件的 SVN 中产生的日志,可以有效地还原一个较粗的过程模型.之所以产生一个较粗的模型,原因在于软件过程实施的过程实例相对较少,本文将这种现象称为单实例性.

软件过程的单实例性是指软件过程在实施的过程中所产生的过程实例较少,往往形成只包含一个案例的轨迹.单实例性是软件过程的基本特征.比如,在生产软件 A 时所使用的过程,不会再一成不变地指导软件 B 的开发.也就是生产不同软件所使用的软件开发过程往往是存在差异的,这样就会导致一个过程模型无法像业务过程一样产生大量的过程实例.

但是,单实例性会对过程挖掘方法在软件过程领域的实施提出挑战.当前,过程挖掘方法主要是针对业务过程挖掘提出的.业务过程挖掘被广泛研究和应用[33−35],传统业务过程挖掘理论与方法认为:过程是由多个案例组成的,一个案例是由多个事件组成的,事件日志的案例信息对传统过程挖掘而言是必不可缺的[9].但是软件过程挖掘面临的困难往往是过程实例只有一条(本文将这种只包含一条案例的轨迹称为单触发序列),如何通过软件过程的单触发序列来发现软件过程模型,是本文面临的重要挑战之一(本文将该问题称为单实例性问题).

针对该问题,文献[7]提出了一种增量式的流程挖掘方法,并尝试将该方法集成到过程挖掘框架中来,从SCM 数据中获取软件过程模型.该文利用同一个开源软件 ArgoUML 的 5 种不同编程语言所采集的数据进行挖掘,该方法的思想是,一旦有其他数据产生,就可以重新细化原来的过程模型.这样,随着日志数量的增加,过程模型的精细程度也就越来越高.该方法并不能很好地适用于所有软件过程的发现,因为并不是所有软件开发都存在多个开发实例.文献[8]与文献[36]分别对软件系统本身的处理过程进行挖掘,前者提出了一种自底向上的方法,该方法利用一个软件系统的事件日志(比如轨迹数据)来分析用户和系统的运行时行为来改进软件;后者对分布式系统进行研究.

尽管单实例性问题导致过程挖掘算法不能很好地适用于软件过程,但是通过软件过程的另一重要特性——迭代性,能够对软件过程挖掘的单实例性问题加以解决.

2 基本表述

软件过程挖掘涉及模型与数据的表达,模型用来描述挖掘出的过程模型;数据为过程挖掘的入口,原始数据不能用于直接挖掘,需要先对数据进行抽取以支持过程挖掘算法的实施.因此,下面将分别对软件过程模型、软件过程数据以及软件过程日志的相关概念进行表述.

2.1 软件过程模型

当前,实践中已经存在了大量过程建模语言,包括标号迁移系统、业务过程建模与标记(BPMN)、Petri 网、工作流网、YAWL、事件驱动过程链、因果网以及 UML 活动图等.这些建模语言间含有大量共性的内容(比如大多包含两类节点:活动和控制),甚至其中有些模型能够相互转化(比如 Petri 网和 BPMN 模型[37]、变迁系统[38]或 UML 活动图模型[39]的转换等).为此,本文并不局限于某种建模语言,而采用基本 Petri 网系统(EN 系统)对本文的基本方法和原理进行说明.这是因为 Petri 网具有严格的数学表示,且应用最为广泛.同时,在本文作者已提出的软件演化过程元模型 EPMM[15]的基础上,给出软件过程的相关形式化定义.

定义 1(软件过程)[15]. 一个四元组 $p=(C,A;F,M_0)$为一个软件过程,其中,

© 中国科学院软件研究所 http://www.jos.org.cn

(1) $(C,A;F)$是一个没有孤立元素的 Petri 网,$A\cup C\neq\varnothing$.

(2) C 是一个条件的有限集;$\forall c\in C$ 称作一个条件.

(3) A 是一个活动的有限集;$\forall a\in A$ 称作一个活动,A 中元素 a 的发生称为 a 被执行或者 a 点火.

(4) $M_0(M_0\subseteq C)$为 p 的初始标记,一个 $d\in M_0$ 被称为托肯.

定义 2(活动)[15]. 一个活动为四元组 $a=(I,O,L,B)$,其中,

(1) I,O,L 分别为输入数据结构、输出数据结构以及本地数据结构,这些数据结构被当作资源的抽象.

(2) B 为活动体,既可以为一个软件开发过程 p,也可以为一系列功能 F 的集合.一个功能 F 的主体为一个二断言 $A(F)=(\{Q_1\},\{Q_2\})$,其中,Q_1,Q_2 为一阶谓词公式,Q_1 为前条件定义了活动 a 执行前的状态,Q_2 为后条件定义了活动 a 执行后的状态.

定义 3(使能、后继情态)[40]. 设 $p=(C,A;F,M_0)$为一个过程模型,则:

(1) 设 $a\in A$ 且 $c\subseteq C$,a 是情态 c 使能(c-enabled)的(或称为 a 在情态 c 有发生权)当且仅当${}^\bullet a\subseteq c\wedge a^\bullet\cap c=\varnothing$.

(2) 设 $a\in A,c\subseteq C$,且 a 是 c 使能的.新情态 $c'=(c-{}^\bullet a)\cup a^\bullet$是 c 在 a 上的后继情态(follower case),记为 $c[a\rangle c'$. 有时为了方便,在不引起混淆的情况下,$c[a\rangle c'$也可简记为$[a\rangle c'$或者 $c[a\rangle$或者$[a\rangle$.

本文使用基本 Petri 网系统(EN 系统)和 EPMM 对软件过程进行描述,文献[40,41]对 EN 系统的相关概念进行了详细解释,EPMM 可参阅文献[15],此处不再赘述.

2.2 软件过程数据

软件过程数据涵盖了软件过程中出现的所有数据,这些数据被各种数据管理系统进行存储和管理.事实上,当前绝大多数软件密集型企业都有利用某种数据记录和管理软件来对他们项目实施开展中产生的数据进行存储,以构建他们自身的软件过程数据管理系统.当前,主流的软件过程数据管理系统包括 SCM,PMS,DTS 等.

图 2 为软件过程数据与软件过程挖掘的关系.随着软件产品的开发和维护,大量包含与开发活动相关的过程实施数据被产生.这些数据被各个数据管理系统所存储,通过将这些系统中的数据抽取,能够进行软件过程的挖掘,挖掘出过程模型.软件开发过程挖掘的主要作用包括过程发现、过程改进以及过程监控.在以过程为中心的软件工程环境中,软件过程模型的建立越来越困难,过程工程师当前的工作应该更偏重于通过已有的过程数据进行过程发现,而非将大量精力集中在过程模型的建立.为此,使用过程挖掘的方法对软件过程模型发现将是软件过程发展的又一重要趋势.

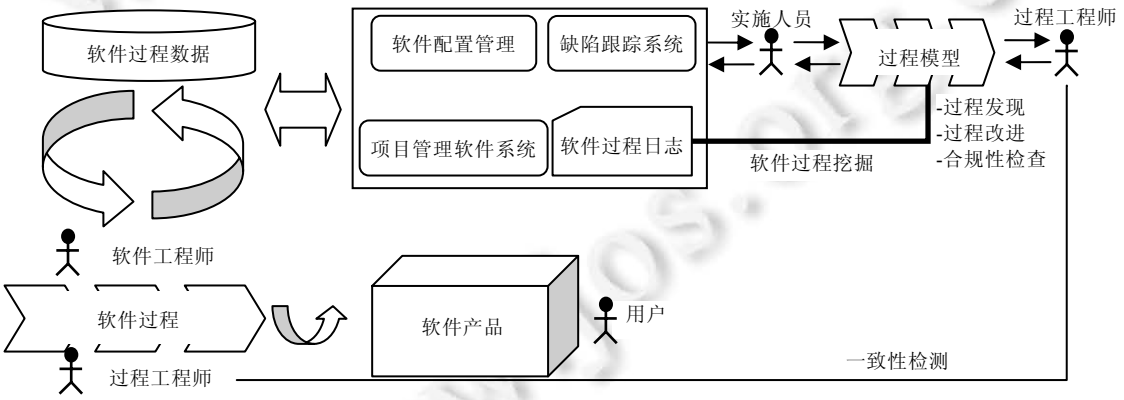


Fig.2 Software process data and software process mining

图 2 软件过程数据与软件过程挖掘

软件过程数据是软件过程执行后留下的“痕迹”,通过对于过程数据的分析与挖掘,就可以还原一个真实的开发过程模型.这样的模型比通过过程工程师设计的模型更加精确、更加接近真实,并且更加快速,软件过程挖掘为软件过程数据分析与过程管理之间架起一座桥梁.

© 中国科学院软件研究所 http://www.jos.org.cn

作为本文挖掘的基础,软件过程数据起着决定性的作用,没有数据也就没有过程挖掘,本文主要以软件配置管理系统 SCM 中的数据为基础,对这些数据进行视图的统一,提出软件开发过程日志的概念,为后续的挖掘工作奠定基础.

SCM 能够被用于整个软件过程的生命周期,目的是对变更进行标识、控制以及确保变更确实实现,并向相关人员进行报告.它提供工作空间管理、并行开发支持、过程管理、权限控制、变更管理等一系列全面的管理能力,已经形成了一个完整的理论体系.同时,在软件配置管理的工具方面也出现了大批产品,如最著名的 IBM 公司的 ClearCase(http://www-03.ibm.com/software/products/zh/clearcase)、有将近 20 年历史的 Perforce(https://www.perforce.com/)、开源产品 CVS(http://www.nongnu.org/cvs/)、Microsoft 公司的 Visual Source Safe(https://msdn.microsoft.com/en-us/vstudio/aa718670.aspx)、Apache 公司的 SVN(http://subversion.apache.org/)以及分布式存储管理系统 GitHub(https://github.com/)等.也就是说,SCM 是一种标识、组织和控制修改的技术,目的是使错误降为最小并最有效地提高生产效率.从功能角度,SCM 能够概括为 3 个方面的作用:版本控制、变更控制以及过程支持.大多数版本控制软件都能对软件系统的开发进程进行详细的记录,对整个软件生命周期中所产生的数据进行存储的目的不仅仅是用于帮助企业对软件项目进度进行管理,还包括对软件过程进行复审、对软件实施中存在的问题进行追踪和重现等.

当前,SCM 系统逐渐从 SVN,CVS 等经典的集中式管理向分布式存储管理系统 GitHub 进行改变,它相对于传统集中式管理 SVN,CVS 有着以下优势:注重社会协作、支持离线、提交为原子级、每个工作树包含完整项目历史仓库等.但是无论是集中式还是分布式,对于软件版本的管理、开发过程中的数据存储,软件的主要功能和目的均没有太大改变,这些数据抽象后,其挖掘的方法原理是一致的.因此,本文以 SCM 系统中经典的 SVN 数据为切入口,对比传统事件日志与软件开发过程日志间的区别与差异,进而提出软件过程日志的形式化表达.

2.3 软件过程日志

过程挖掘的相关概念主要是围绕如何表达事件日志展开的.事件日志是一切过程挖掘的起点,事件日志的形式化符号体系是过程挖掘方法、理论、实践内容的重要基础.下面给出软件过程日志相关定义:

定义 4(软件过程轨迹,软件过程日志). 设 A 为软件过程活动的集合,则一个只含有活动名称的序列 $\sigma\in A^*$ 为一条软件过程轨迹,且 $L\in\mathcal{P}(A^*)$ 为一个软件过程日志,其中,$\mathcal{P}(A^*)$ 为 A^* 的幂集.

软件开发过程日志 L 在不引起混淆的情况下简称为过程日志;同样的,软件开发过程轨迹 σ 能够被简称为过程轨迹.该定义是从活动的角度进行定义,而现实情况是在过程数据中不存在活动和案例信息,仅仅包含事件.

事件日志中包含了整个事件的大量信息,这些信息涵盖了事件的方方面面,但是在过程挖掘的理论方面,大多数时候我们只关心活动、案例、事件等的名称,因此为了简化事件日志,下面给出简单事件日志的概念.

定义 5(简单过程日志). 简单过程日志 L 是轨迹的集合,它被定义为 A 上轨迹的一个多集(记为 B),即 $L\in B(A^*)$.

例如:$L_1=[\langle a,b,c,d\rangle^4,\langle a,e,f,g\rangle^8,\langle a,e,d\rangle]$,通过该例子可以看出,简单过程日志 L_1 包含 13 个轨迹,其中,轨迹 $\langle a,b,c,d\rangle$ 发生了 4 次,轨迹 $\langle a,e,f,g\rangle$ 发生了 8 次,轨迹 $\langle a,e,d\rangle$ 发生了 1 次.在简单过程日志中,不含有任何属性,即便是时间戳和资源信息都被省略,事件从左至右的顺序代表了事件发生的先后顺序.

定义 6(软件过程事件,属性). 设 E 为软件过程事件空间,即所有可能的事件标识符的集合.事件由不同的属性来描述.设 AN 为属性名的集合.对于任意事件 $le\in E$ 且属性名 $an\in AN$,则 $\#_{an}(le)$ 表示事件 le 的属性 an 的值.如果事件 le 不含有属性 an,则 $\#_{an}(le)=\perp$(空值).

软件过程事件属性是从大量过程数据中抽象所得到的,属性的确定以是否能够有效支持后续挖掘工作为基本标准.本文定义软件过程事件 le 的属性集 $AN=\{id,date,paths,msg\}$.*id* 属性是事件的唯一标识,*date* 属性是事件发生的时间,*paths* 属性是开发活动具体影响的那些文件,*msg* 属性是事件相关的描述信息.这些属性间仍然能够嵌套属性,比如,*paths* 属性是由多个 *path* 属性组合而成,而 *path* 又是由 *action* 和 *address* 组成,其中,*action* 表示对文件操作具体是增加、删除还是修改,*address* 属性是具体的路径.

下面就以 SVN 日志数据为例,对传统事件日志与软件开发过程日志间的区别进行讨论.软件开发过程数据

© 中国科学院软件研究所 http://www.jos.org.cn

中的数据没有明确地给出用于过程挖掘的关键属性信息:活动属性以及案例属性等.SVN是一种典型的SCM系统,其日志数据是在软件开发过程中开发活动的行为记录,通过SVN日志数据能够对软件开发过程进行分析与挖掘,SVN日志能够以XML文档的形式进行导出,这些日志信息主要包括了版本号(revision)、作者(author)、行为(action).

图3为一个只包含3个事件记录的SVN日志文档,记为svnFile.可以看出,svnFile未包含任何案例信息、未关联任何活动信息.传统事件日志与SVN日志数据间结构对比如图4所示.

```
⟨log⟩
  ⟨logentry revision="23823"⟩
    ⟨author⟩ezust⟨/author⟩
    ⟨date⟩2015-01-04T14:43:40.032552Z⟨/date⟩
    ⟨paths⟩
      ⟨path text-mods="true" kind="file" action="M" prop-mods="false"⟩/jEdit/trunk/org/gjt/sp/jedit/jedit.props⟨/path⟩
    ⟨/paths⟩
    ⟨msg⟩add new property to jedit.props so it is documented somewhere.⟨/msg⟩
  ⟨/logentry⟩
  ⟨logentry revision="23882"⟩
    ⟨author⟩kerik-sf⟨/author⟩
    ⟨date⟩2015-02-27T08:47:59.670471Z⟨/date⟩
    ⟨paths⟩
      ⟨path prop-mods="false" text-mods="true" kind="file" action="M"⟩
      /jEdit/trunk/org/gjt/sp/jedit/bufferset/BufferSetManager.java⟨/path⟩
      ⟨path prop-mods="false" text-mods="true" kind="file" action="M"⟩/jEdit/trunk/org/gjt/sp/jedit/View.java⟨/path⟩
    ⟨/paths⟩
    ⟨msg⟩documentation only (BufferSetManager) ⟨/msg⟩
  ⟨/logentry⟩
  ⟨logentry revision="23821"⟩
    ⟨author⟩ezust⟨/author⟩
    ⟨date⟩2015-01-04T14:31:09.837049Z⟨/date⟩
    ⟨paths⟩
      ⟨path text-mods="true" kind="file" action="M" rop-mods="false"⟩/jEdit/trunk/org/gjt/sp/jedit/EditPane.java⟨/path⟩
    ⟨/paths⟩
    ⟨msg⟩fix the documentation jEdit.setProperty to add some properties ⟨/msg⟩
  ⟨/logentry⟩
⟨/log⟩
```

Fig.3 A simple SVN log file

图3 一个简单的SVN日志文档svnFile

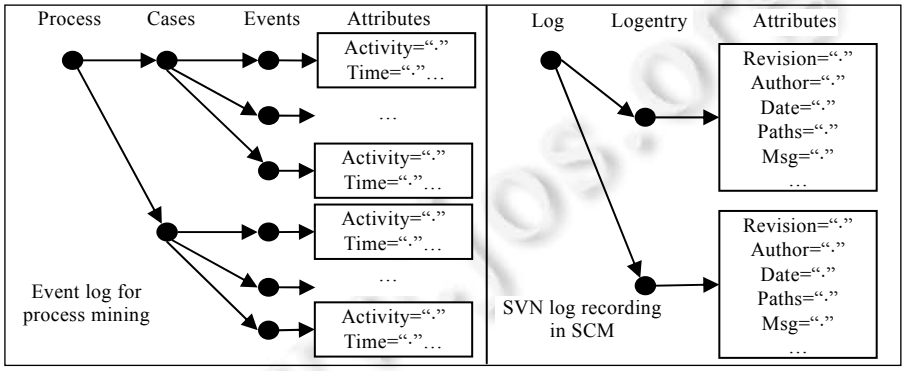


Fig.4 Similarities and differences between event log and SVN log

图4 事件日志与SVN日志间的异同

使用软件开发过程日志对svnFile进行形式化的描述.svnFile中的Logentry元素可以被认为是一个软件开发过程事件,不妨设svnFile中的3个事件为le_1,le_2,le_3,则$\#_{revision}(le_1)$="23823",$\#_{date}(le_1)$="2015-01-04T14:43:

© 中国科学院软件研究所 http://www.jos.org.cn

40.032552Z",$\#_{paths}(le_1)$={(M,"/jEdit/trunk/org/gjt/sp/jedit/jedit.props")},$\#_{msg}(le_1)$="add new property to jedit.props so it is documented somewhere.".同理能够对 le_2,le_3 进行表示,由于 *revision* 属性具有唯一性,因此也可以将其作为 *id* 属性.

通过从软件开发过程数据向过程日志的抽象,能够对数据进行过滤,去除对于挖掘没有意义的数据和属性,使得这些来自不同的 SCM 系统的数据能够形成一个统一的过程日志视图.每个过程事件本质上是活动的实例,因此每个过程事件都能够被关联到某个活动上.下面对过程事件的活动关联进行定义,方便下一节活动层挖掘使用.

定义 7(事件与活动关联关系). 事件 *le* 与活动 *a* 的关联关系记为 *le* R *a*,它是事件日志空间 *E* 与软件活动集 *A* 上的卡氏集 $E\times A$ 的子集,即 *le* R $a\in\{\langle le,a\rangle|le\in E\wedge a\in A\}$,且关联函数 R(*le*)=*a*.若存在多个事件 le_1,le_2,le_3 与活动 *a* 关联,则记为 $\{le_1,le_2,le_3\}$ R *a*.

定义 8(活动分类器). 设 *L* 为一个过程日志,*A* 为 *L* 上活动的集合,过程事件与活动间的关联关系为 R,则事件序列$\langle le_1,le_2,\ldots,le_n\rangle$被转化为轨迹$\sigma=\langle R(le_1),R(le_2),\ldots,R(le_n)\rangle$.

例如,设存在事件序列$\langle le_1,le_2,le_3,le_4,le_5,le_6,le_7\rangle$,其中,$\{le_1,le_2,le_3\}$ R *a*,le_4 R *b*,$\{le_5,le_6\}$ R *c*,le_7 R *d*,则该事件序列通过活动分类器就可以转化为轨迹$\langle a,a,a,b,c,c,d\rangle$.

定义 9(单触发序列)[42]. 设活动集为 *A*,且过程日志 $L=[\sigma](|L|=1)$,其中,轨迹$\sigma\in A^*$,A^*为 *A* 上的所有有限序列的集合.这样的轨迹是一组活动名称的序列(比如$\sigma=\langle a_1,a_2,a_3,\ldots,a_n\rangle$,其中,对于 $1\leqslant i\leqslant n$ 有 $a_i=\sigma(i)$表示轨迹σ的位置 *i* 的活动为 a_i,$|\sigma|=n$ 表示轨迹的长度),当$|\sigma|>|A|$时,将轨迹σ称为单触发序列.

3 双层次软件过程挖掘框架

本文对所使用的基本概念进行了定义,下面将对软件过程挖掘展开讨论.挖掘工作是一项相对复杂的任务,需要进行系统的规划,因此,本文提出一种双层次的软件过程挖掘框架.

李在文献[15]提出的 EPMM 中自顶向下定义了软件演化过程的 4 个层次:全局层、过程层、活动层以及任务层,其中,活动层是对软件演化过程的活动进行建模,从活动的视角进行研究;而过程层是对软件演化过程的过程进行建模,从过程层面进行分析.该思想符合 CMMI 对于软件过程的定义,本文也借鉴该思想将软件过程挖掘分为活动层挖掘和过程层挖掘,活动层挖掘即关注如何从过程日志中发现活动;过程层挖掘关注如何从活动信息中进一步发现过程模型.

本文提出的双层次的软件过程挖掘框架的基本结构如图 5 所示.

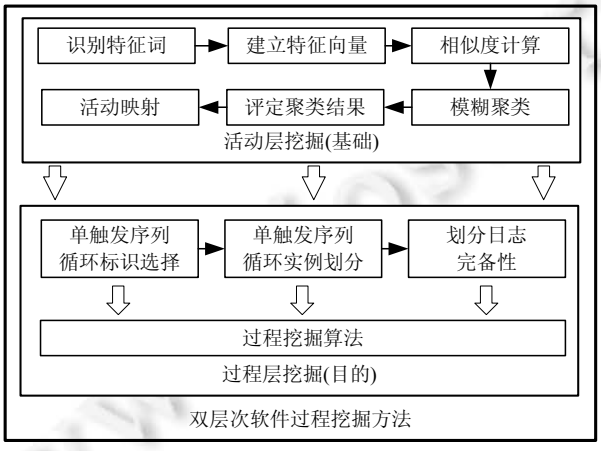


Fig.5 Basic structure of the software process mining framework

图 5 软件过程挖掘框架的基本结构

© 中国科学院软件研究所 http://www.jos.org.cn

软件过程挖掘框架分为活动层与过程层挖掘,活动层挖掘是过程层挖掘的基础,过程层挖掘是活动层挖掘的目的.在活动层,首先对输入的过程日志中的事件进行以下处理:对每个事件中的特征词进行识别、通过特征词建立每个事件的特征向量、计算事件间的相似度、利用模糊聚类对事件进行聚类、对聚类结果进行评价以及完成最后事件与活动的关联.活动层挖掘主要目标是从过程日志中发现活动信息,其输入是过程日志,输出是一条由日志中每个事件关联的活动所形成的单触发序列,该序列作为过程层的输入,为过程层挖掘奠定基础.

过程层挖掘是对由活动形成的单触发序列进行研究,其基本思路是对单触发序列中的循环进行发现,并将每条循环实例作为案例信息进行过程挖掘.过程层挖掘的目的是为了将由活动形成的单触发序列进行循环实例的划分,因此包括对循环标识进行发现、根据循环标识进行循环实例划分以及对划分日志的完备性进行讨论等.划分后能够使用现有的过程挖掘算法对循环进行挖掘,最后再将结果进行整合.

4 活动层挖掘

活动层挖掘的目的是从过程日志中发现每个事件所关联的活动信息.首先对过程日志中的事件进行分析,抽取每个事件所对应的特征词,然后通过加权结构连接向量模型将每个事件转化为一个特征向量,进而通过特征向量间的相似性能够度量事件间的相似程度,最后对过程日志中的事件进行模糊聚类,并基于信息熵提出一种平均活动熵的方法作为最终的聚类结果,该聚类结果即为每个事件所对应的活动.

过程日志的本质是活动执行的“痕迹”,事件是活动实例的记录.每个事件反映了一个活动具体在“做什么”.通过对事件进行聚类,能够从蕴含在事件中的语义信息来发现活动.这样,每个事件所关联的活动信息就能够被发现,为下一步的过程层挖掘奠定基础.

4.1 特征词抽取

软件过程日志中,每个事件 *le* 的属性集 *AN*={*id*,*date*,*paths*,*msg*}.*id* 属性被用于唯一的标识该事件,*date* 属性表达事件发生的时间,*paths* 属性表达了事件所涉及的文件的路径,而 *msg* 属性用于抽取该事件的语义信息.由于 *paths* 属性蕴含了对文件操作的具体路径,*msg* 属性蕴含了对操作的描述,因此通过将 *paths* 属性和 *msg* 属性转化为特征向量,进而对事件间的相似性进行计算.

过程日志中的事件是包含有事件的隐式语义信息的,换言之,我们无法直接获取每个事件到底做了什么,但是哪些事件是在做相关的事情,哪些事件是做不同的事情是能够被发现的.因此,可以将最相关的事件通过聚类的方法关联到相同的活动,而将不同的事件关联为不同的活动.为了发现事件间的相关性,需要先对过程日志中每个事件的属性 *paths* 和 *msg* 进行特征词抽取.抽取出的特征词将形成特征词集,定义如下:

定义 10(特征词集合). 用符号$\$_{an}(le)$表示事件 *le* 的属性 *an* 对应的特征词的集合,$|\$_{an}(le)|$表示 *an* 属性所对应特征词集合中元素的个数.

例如,$\$_{revision}(le)=\varnothing$,$\$_{msg}(le)=\{add,property,jedit,prop,document\}$,$|\$_{msg}(le)|=5$.

过程日志中的属性 *paths* 中为具体路径信息,能够使用正则表达式直接分离.属性 *msg* 中的信息为自然语言所描述,本文考虑使用斯坦福 CoreNLP(http://nlp.stanford.edu/software/corenlp.shtml)进行处理.在特征词抽取之前需要说明的是,1) 属性 *msg* 信息只用英语进行描述(也就是本文只考虑英文信息);2) 属性 *msg* 信息一定是规范的、与该条日志密切相关的.

4.2 加权结构连接向量模型

本文为了区别一些特征词的重要程度,为每个特征词赋予相应的权值,在基于 TF-IDF 思想的传统结构连接向量模型(structured link vector model,简称 SLVM)[43,44]的基础上进行改进,提出了一种加权结构连接向量模型(weighted structured linked vector model,简称 WSLVM).一些基于与结构相关联的方法[45−48]的本质是将每个 XML 文件的结构层次量化作为其内容的权值,而对于过程日志而言其结构是明确的,且由于不同属性的重要程度是不同的,比如,若 *paths* 比 *msg* 属性更加重要,其权值也应该高于 *msg* 属性.为了能够对属性的重要程度进行度量,提出属性权值的概念.

© 中国科学院软件研究所 http://www.jos.org.cn

定义 11(属性权值). 用符号$\alpha_{an}(le)$表示事件 *le* 的属性 *an* 的权值.

为每个特征词设置权值的好处在于:1) 每个特征词的属性不同其重要程度也应该不同;2) 可以减少不同属性间相同的两个特征词的相关性;3) 可以有意识的控制某个属性的重要性,比如可能在某些情况下,属性 *msg* 比属性 *path* 更加重要.另外,如果某个特征词在多个属性间出现应该分别计算.例如,对于事件 le_1 中属性 *msg* 的权值为 0.8,即$\alpha_{msg}(le_1)=0.8$,$\$_{msg}(le_1)=\{a,b,c,d\}$,则事件中每个特征词 *a*,*b*,*c*,*d* 的权值均为 0.2;此时,如果其他条件不变的情况下,若 le_2 中$\alpha_{path}(le_2)=0.2$,$\$_{path}(le_2)=\{a,e,f,g\}$,则事件中每个特征词 *a*,*e*,*f*,*g* 的权值均为 0.05.这样,尽管两个事件中出现了特征词 *a*,但是属性 *msg* 的 *a* 要比属性 *path* 的 *a* 更加重要.下面给出 WSLVM 的具体定义:

定义 12(WSLVM). WSLVM 是将一个过程日志 *Log* 使用一个矩阵$\Delta_x \in R^{m\times n}$进行表达,表达方式如下:

$$\Delta_x=[\Delta_{x(1)},\Delta_{x(2)},\ldots,\Delta_{x(m)}],$$

其中,*m* 为过程日志 *Log* 中事件的数量,*n* 表示过程日志 *Log* 中所有特征词的数量,设 le_x 来表示日志 *Log* 中第 *x* 个事件且$\Delta_{x(i)}\in R^m$是指 le_i 的特征向量,则有:

$$\Delta_{x(i,j)}=\alpha_{(j)}TF(w_j,le_i)IDF(w_j),1\leqslant i\leqslant m,1\leqslant j\leqslant n,$$

$$\alpha_x=[\alpha_{(1)},\alpha_{(2)},...,\alpha_{(n)}],\sum\nolimits_i\alpha_i=1,$$

$$\alpha_{(j)}=\Sigma_{\forall an\in AN\wedge x=le_j}\alpha_{an}(x)\ /|\$_{an}(x)|,$$

其中,$F(w_j,le_i)$是指在事件 le_i 中特征词 w_j 出现的频率;$DF(w_i)=\log(m/DF(w_j))$是特征词 w_j 的逆事件频率,来消减频繁出现特征词的重要性;$F(w_j)$为包含有特征词 w_j 的事件的数量;$\alpha_{(j)}$为每个特征词的权重,其中,*an* 为该特征词所对应的属性.为了减少由于事件内容变化大小的影响,我们引入归一化.

定义 13(特征向量归一化). 归一化的事件特征向量定义为

$$\bar{\Delta}_{x(i,j)}=\Delta_x(i,j)/\|\Delta_x(i)\|_2,$$

其中,$\|\Delta_x(i)\|_2$是指$\Delta_x(i)$的 2 范数.

通过 WSLVM 方法,每条过程日志中的事件就可以被转化为特征向量,整个过程日志被转化为一个加权特征矩阵.例如,图 3 中的简单 SVN 日志利用 WSLVM 方法能够进行表达(表 1),该日志对应的特征向量Δ_x,归一化的文档特征向量 $\bar{\Delta}_x$ 分别表示为 0 和表 2.其中,权值的定义分别为$\alpha_{msg}(le)=0.5$,$\alpha_{paths}(le)=0.3$,$\alpha_{action}(le)=0.2$.

需要说明的是,由于 TF-IDF 是一种抑制事件内无意义高频词的负面影响的算法,比如本例中每个事件均含有属性为 *action* 的特征词 M,因此所对应的 IDF 的值均为 0.这是因为如果一个特征词在所有文档中均出现,它是无法用来区分不同的事件的.

Table 1 Document feature vector corresponding to the simple SVN log in Fig.3

表 1 图 3 中简单 SVN 日志对应的文档特征向量

特征词	Logentry		
	23823	23882	23821
add	0.003 1	0	0.002 4
property	0.003 1	0	0.002 4
jedit	0	0	0
prop	0.008 5	0	0
document	0.008 5	0	0
M	0	0	0
trunk	0	0	0
org	0	0	0
gjt	0	0	0
sp	0	0	0
jedit.props	0.003 6	0	0
documentation	0	0.005 3	0.002 4
buffersetmanager	0	0.014 5	0
bufferset	0	0.074 4	0
buffersetmanager.java	0	0.074 4	0
view.java	0	0.074 4	0
fix	0	0	0.611 6
setproperty	0	0	0.611 6
editpane.java	0	0	0.314 6

© 中国科学院软件研究所 http://www.jos.org.cn

Table 2 Normalized document feature vector corresponding to the simple SVN log in Fig.3

表 2 图 3 中简单 SVN 日志对应的归一化的文档特征向量

特征词	Logentry		
	23823	23882	23821
add	0.235 5	0	0.225 7
property	0.235 5	0	0.225 7
jedit	0	0	0
prop	0.638 1	0	0
document	0.638 1	0	0
M	0	0	0
trunk	0	0	0
org	0	0	0
gjt	0	0	0
sp	0	0	0
jedit.props	0.273 5	0	0
documentation	0	0.343 4	0.225 7
buffersetmanager	0	0.930 3	0
bufferset	0	0.001 2	0
buffersetmanager.java	0	0.001 2	0
view.java	0	0.001 2	0
fix	0	0	0.006 5
setproperty	0	0	0.006 5
editpane.java	0	0	0.003 4

通过将整个过程日志量化为一个特征矩阵,矩阵的每一行为一个事件所对应的特征向量;同时,为了能够有效控制每种属性对结果的影响,我们加入属性权值,即,每个属性都对应一个权值,根据其属性不同来有意识的调控不同属性间的重要程度.

量化后,特征向量的相似性主要通过向量距离来衡量.一般常用的方法是,通过向量间的余弦值进行计算.

4.3 活动发现及关联

每个事件被向量化后,就能够对过程日志中的相似性进行度量,再通过某种聚类方法,就可以将相似性较强的事件关联为一类,而将相似性较弱的事件区分开.该方法好处在于:能够将一些原本看似无关的事件按照事件语义进行划分,进而得到做“同一件事”的一类事件.在传统过程挖掘中,活动信息是不可或缺的,但是现实情况是,某些非面向过程的 SCM 系统并不会记录任何活动信息,因此,文献[9]中介绍了一种过程挖掘领域中常用的通过多个属性来确定活动的分类器.分类器在业务过程领域取得了较好的应用,但由于在开发过程中的单实例性以及过程数据的记录并非面向过程的,故通过分类器的方法从事件发现活动是不可能的.然而,为了有效地发现事件所关联的活动,本节在上一节的基础上,首先利用一种典型的模糊聚类的方法——模糊 C 均值聚类算法 FCM(fuzzy C-means)[49]来确定聚类的活动个数、聚类中心以及相应的隶属度矩阵;其次,提出一种基于 H_{Pal} 熵的确定聚类结果的方法;最后,通过聚类的结果将事件与发现的活动相关联,完成软件过程活动发现.

确定聚类的活动个数、聚类中心以及相应的隶属度矩阵的基本思想是:设在软件过程日志中事件的个数为 LN,通过模糊聚类的方法将事件关联的活动数为 $C(2\leqslant C\leqslant LN)$,其对应的隶属度矩阵集合记为 M_U,聚类中心集合记为 $Center$.对输入的由事件构建所得特征矩阵 Δ_x,根据活动数 C,使用 FCM 对其进行聚类,当满足阈值 ε 或者达到最大迭代次数 K 时迭代停止,记录下取 C 时所对应的隶属度矩阵和聚类中心,最后输出所有活动数 C 所构建的隶属度矩阵集合 M_U 和聚类中心集合 $Center$.该隶属度矩阵和聚类中心为下一步的划分确定奠定了基础.

本文为了从过程日志中获取活动个数,考虑通过聚类个数变化,通过活动熵来评估哪种聚类结果更优,进而确定聚类个数.信息熵被用来度量信源发送信息能力大小.经典信息论中,信息熵的定义如下:对于离散无记忆信源 X,如果其概率空间为 $[X,P]=[x_k,p_k|k=1,2,...,N]$,则信源 X 的平均不确定性记为 $H(X)$,即为信源 X 的熵,其计算公式为 $H(X)=\sum_k p_k \log 1/p_k$. 可以看出,对于 p_k 而言,一般是不能为 0 的.如果存在为 0 情况的解决方案一般有两种:要么在计算的时候对这种情况单独考察,另一种则是引入一种新的计算方案.而本文采用了后者,引入 H_{Pal} 熵作为评价模糊聚类结果.H_{Pal} 熵是由 Pal 等人在文献[50]提出并成功地应用于图像分割处理的,大量文献证明了 H_{Pal} 熵的正确性及实用性[51,52].同时,通过本文实验部分也可以看出,H_{Pal} 熵有效地避免了单独考察引起的不

© 中国科学院软件研究所 http://www.jos.org.cn

必要的逻辑判断,增加了程序的效率.

定义 14(H_{Pal}熵)[50]. $H=\sum_{i=1}^{n} p_i e^{1-p_i}$.

定义 15(平均活动熵). 给定过程活动数为 C,对应隶属度矩阵为 U,则平均活动熵定义为 $H(C)=\frac{1}{N}\sum_{i=1}^{C}\sum_{j=1}^{N}\mu_{ij}e^{1-\mu_{ij}}$,其中,$\mu_{ij}$表示样本 j 属于类 i 的程度.

通过定义 15 可知,针对不同 C 的平均活动熵 $H(C)$也不同.在所有的活动数 $C(2\leqslant C\leqslant LN)$中,存在一个活动数 m,使得平均活动熵 $H(m)$达到最小值.此时,所对应的活动数 m 即为最佳聚类结果,对应的隶属度矩阵记为 U_m 及聚类中心记为 CT_m.具体如算法 1 所示.

算法 1. 聚类结果确定及活动关联.

输入:隶属度矩阵的集合 $M_U=\{U_1,\ldots,U_C\}$,聚类中心集合 $Center=\{CT_1,\ldots,CT_C\}$,隶属度阈值$\delta$.

输出:活动集 A 以及在事件和活动集 A 上的关联关系集 R.

步骤 1:根据活动数 C 和相应的隶属度矩阵依次计算每个 C 所对应的活动熵,得到活动熵集合:

$$Activity\_Entropy=\{activity\_entropy_1,\ldots,activity\_entropy_C\}.$$

步骤 2:对活动熵集合归一化,得到归一化的活动熵集合:

$$Activity\_Entropy'=\{activity\_entropy/e^{1-1/c},\ldots,agent\_entropy/e^{1-1/c}\}.$$

步骤 3:得到 $C_i=\arg\min_i Activity\_entropy'$ 及 C_i 所对应的聚类中心 $Center_i$、隶属度矩阵 U_i.

步骤 4:根据隶属度阈值δ、聚类中心 $Center_i$、隶属度矩阵 U_i 得到当前每个元素的聚类集合,也就是活动集 A;以及 A 与各个元素的相关关系,也就是活动集 A 上的关联关系集 R.

步骤 5:输出活动集 A 以及在事件和活动集 A 上的关联关系集 R.

5 过程层挖掘

通过上一节对软件过程事件与发现的活动进行关联,活动层挖掘的结果将是一条单触发序列,它又作为过程层挖掘的输入,利用过程层挖掘方法,最终挖掘出软件过程模型.因此,单触发序列挖掘是过程层挖掘的重点.针对单触发序列的研究,文献[42]提出一种基于启发式方法的单触发序列挖掘算法,该算法的思想是:将通过对单触发序列中的循环进行划分,从而形成多条循环实例;基于多条循环实例再利用已有的过程挖掘算法,最终能够将循环进行精确挖掘.但是该文只针对全部循环(模型本身就是循环)的情况进行讨论,而未对非完全循环的情况进行论述.另外,循环实例划分的日志完备性也需要被讨论.综上,本节将首先对单触发序列挖掘的基本思路及启发式关系度量方法进行阐述,然后对非完全循环情况以及日志完备性进行讨论.

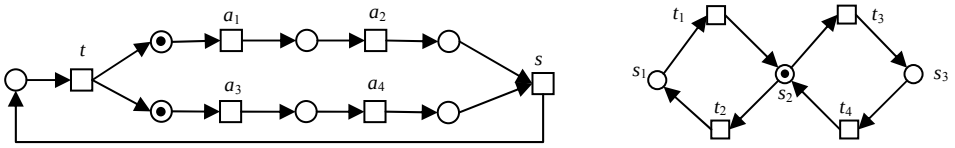
5.1 基本思路及启发式关系度量

传统的过程挖掘可以理解为从一组轨迹中发现过程模型,而单触发序列划分的本质是从一条轨迹中发现属于不同组的事件序列.当前已经存在大量过程挖掘算法,基本上都是假设案例信息的存在,使得这些算法不能很好地支持从单触发序列中挖掘出过程模型.比如,给定活动序列$\langle a,b,c,d,a,c,e,f,g,i,b,a,c,c,c,c,c,p,o,\ldots\rangle$,如何针对单实例的过程轨迹进行挖掘,如何判断哪些活动属于同一个循环实例,哪些活动不属于.

通过将活动根据循环实例进行分组,形成“案例”进而能够发现循环部分对应的过程模型.图 6(a)为某过程模型形成的一条简单的具有并发关系的观察轨迹,可能为$\sigma=\langle a_3,a_4,a_1,a_2,s,t,a_1,a_2,a_3,a_4,s,t,a_3,a_1,a_2,a_4,s,t,a_1,a_3,a_4,a_2,s,t,\ldots\rangle$.该轨迹为能够发现所有行为的完全日志,利用传统过程挖掘方法对该轨迹直接进行挖掘会出现问题.比如:如果使用α算法挖掘结果如图 7(a)所示,α算法主要通过轨迹中活动间的依赖关系进行判断,因此对轨迹的开头非常敏感,图中将模型的初始活动标记为 a_3,而不是将活动 a_1,a_2,a_3,a_4 当作一个整体;同时,α算法无法处理噪声,一旦轨迹中出现噪声,α算法也无法进行处理.使用启发式方法挖掘出的如图 7(b)所示(其中,黑色实心方块表示内部活动不产生任何可观察行为),启发式方法能够处理噪声,但是一旦参数的设置不当,就会导致无法正确地发现活动 a_1,a_2 和 a_3,a_4 之间的并发性.如果能够将本该属于同一循环实例的活动进行划分,然后再进行过程挖

© 中国科学院软件研究所 http://www.jos.org.cn

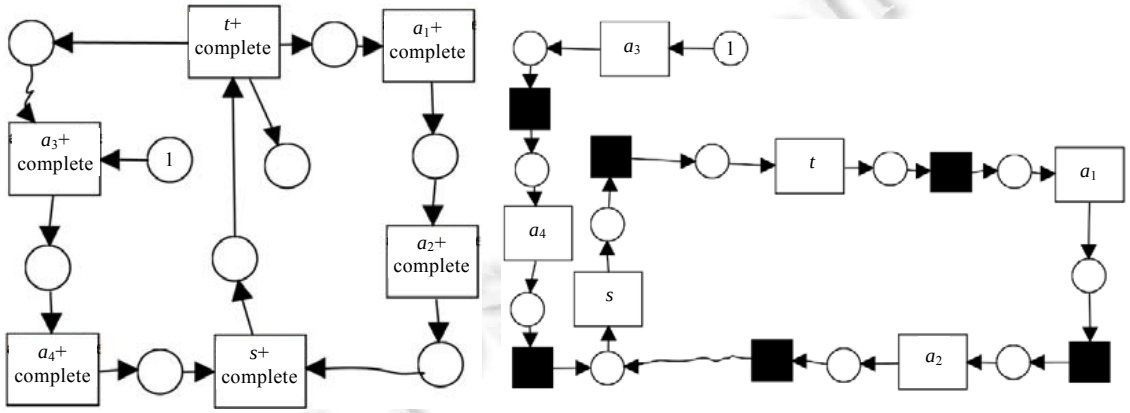
掘,最后再添加循环的话,就能正确地进行挖掘.



(a) (b)

Fig.6 Complete loop example

图 6 完全循环示例



(a) (b)

Fig.7 Single firing sequence mining problem example

图 7 单触发序列挖掘问题图示

不仅仅是并发活动作为轨迹的开头会导致挖掘的失败,其他非顺序结构做为轨迹的开头都无法正确地挖掘出过程模型.导致这些问题的原因在于传统过程挖掘并未对单触发序列中的循环进行处理,而是利用活动间的依赖关系去计算或者推导它们之间的关系,这就会导致挖掘出现问题.但是如果选择活动 s 作为循环标识,则划分后的结果就为 $\{\langle s,t,a_3,a_1,a_2,a_4\rangle,\langle s,t,a_1,a_3,a_4,a_2\rangle,\ldots\}$,通过这样划分,就能够将一条轨迹划分为多个循环实例,如果将每个循环实例作为案例信息进而再利用过程挖掘方法,就能够正确的挖掘出过程模型.类似地,再比如图6(b),如果初始情态为 $\{s_2\}$,则产生的观察轨迹可能为 $\langle t_2,t_1,t_3,t_4,t_3,t_4,t_2,t_1,\ldots\rangle$,这样的轨迹利用传统过程挖掘的知识是无法产生合适的过程模型的.类似的案例还有很多,这里就不再一一赘述,传统过程挖掘需要更多的案例来支撑其挖掘算法的实施,事实上,类似图 6 中的模型尽管只产生一条观察轨迹,某些情况下就足以支持挖掘出整个过程模型,也就是只要能够正确地、合适地将单触发序列中的活动进行循环实例划分,就能够发现整个过程模型.为了进一步对非完全循环情况下单触发序列挖掘进行详细的分析,本文仍基于文献[21,42]中关于循环、循环标识、循环实例及划分,依赖、并发、2-循环度量关系度量的相关定义 16~定义 21,定义内容如下.

定义 16(循环)[42]. 设 $p=(C,A;F,M_0)$为一个软件过程模型,其中,M_0 表示模型的初始状态,使用 $R(M_0)$表示从 M_0 可达的所有情态集合,一个 p 的子网 $LN=(C_L,A_L;F_L,M_e)$为一个循环当且仅当:

(1) $C_L\subseteq C\wedge A_L\subseteq A\wedge F_L\subseteq F\wedge M_e\in R(M_0)$;

(2) 存在一个步序列 $G_0G_1\ldots G_{n-1}(G_0,G_1,\ldots,G_{n-1}\subseteq A_L)$和一组情态 $c_1,c_2,\ldots,c_n\subseteq C$,使得 $M_e[G_0\rangle c_1,c_1[G_1\rangle c_2,\ldots,c_{n-1}[G_{n-1}\rangle c_n$ 且 $c_n\cap M_e\neq\varnothing$,称步序列 $G_0G_1\ldots G_{n-1}$ 为循环步序列.

定义 17(循环标识)[42]. 设子网 $LN=(C_L,A_L;F_L,M_e)$为过程模型 $p=(C,A;F,M_0)$上的循环,若 $M_e[G_0G_1\ldots G_{n-1}\rangle c_n$,用符号 ∂_{set} 表示将序列转换为集合(比如 $\partial_{set}(\langle d,a,a,a,d\rangle)=\{a,d\}$),则循环标识 $L_e\in\partial_{set}(G_0G_1\ldots G_{n-1})\wedge M_e[L_e\rangle\wedge c_n[L_e\rangle$.

© 中国科学院软件研究所 http://www.jos.org.cn

定义 18(循环实例,循环实例划分)[42]. 设单触发序列σ=⟨…a,….,a,…⟩,如果以活动 a 为循环标识,则按照循环标识进行划分,划分结果记为$\sigma(a)$={⟨×a…⟩,⟨×a…⟩,…},其中,$\sigma(a)$中的子序列称为循环实例,且该划分过程称为循环实例划分.其中,×表示可能出现在活动 a 之前的、但是同属于一个循环实例的活动.

定义 19(依赖关系度量)[21]. 设轨迹σ为定义在活动集 A 上的单触发序列,活动 a,b 属于 A,活动 a 与 b 相继发生的次数记为$|a\rangle_\sigma b|$=|{1≤i≤|σ|−1,|$\sigma(i)$=a∧$\sigma(i$+1)=b}|.称

$$a \Rightarrow_\sigma b = \begin{cases} \dfrac{|a\rangle_\sigma b| - |b\rangle_\sigma a|}{|a\rangle_\sigma b| + |b\rangle_\sigma a| + 1}, & \text{if } a \neq b \\ \dfrac{|a\rangle_\sigma a|}{|a\rangle_\sigma a| + 1}, & \text{if } a = b \end{cases}$$

为轨迹σ中活动 a 与活动 b 的依赖关系值.

定义 20(并发关系度量)[42]. 设轨迹σ为定义在活动集 A 上的单触发序列,活动 a,b 属于 A,活动 a 与 b 相继发生的次数记为$|a\rangle_\sigma b|$=|{1≤i≤|σ|−1,|$\sigma(i)$=a∧$\sigma(i$+1)=b}|.称

$$a \|_\sigma b = 1 - \left| \frac{|a\rangle_\sigma b| - |b\rangle_\sigma a|}{|a\rangle_\sigma b| + |b\rangle_\sigma a| + 1} \right| (|a\rangle_\sigma b| > 0 \vee |b\rangle_\sigma a| > 0)$$

为轨迹σ中活动 a 与活动 b 的并发关系值.

定义 21(2-循环度量)[21]. 设轨迹σ为定义在活动集A上的单触发序列,活动 a,b 属于 A,记轨迹σ中序列"aba"的频度为$|a\rangle\rangle_\sigma b|$=|{1≤$i$≤|$\sigma$|−2,|$\sigma(i)$=$a$∧$\sigma(i$+1)=$b$∧$\sigma(i$+2)=$a$}|.称

$$a \Rightarrow_{2\sigma} b = \frac{|a\rangle\rangle_\sigma b| + |b\rangle\rangle_\sigma a|}{|a\rangle\rangle_\sigma b| + |b\rangle\rangle_\sigma a| + 1}$$

为轨迹σ中活动 a 与活动 b 的 2-循环关系值.

5.2 非完全循环情况

文献[42]是在软件过程模型本身就是循环结构的情况下,或者从日志的角度而言,日志中的每个活动都在循环的某个实例中.现实并非完全如此,如可能存在以下软件过程模型,该模型在一个循环的前后都不是循环:

非完全循环下,不处于循环内部的部分会对循环划分产生干扰,并最终导致挖掘出现问题.比如,图 8 中从 t 到 a_1,a_2,a_3,a_4 再到活动 s 是循环结构,在该结构的前边是一个由活动 b 和活动 c 组成的选择结构,在循环结构的后边是一个 h,d,e,f 组成的并发结构.该过程模型可能产生的单触发序列为σ=⟨b,t,a_1,a_2,a_3,a_4,s,…,t,a_3,a_4,a_1,a_2,h,d,e,f⟩,对应的活动集 A={b,t,a_1,a_2,a_3,a_4,s,d,e,f,h},且该模型并不是所有的活动都在循环中,那么这样的日志是否能够使用循环划分的方式进行解决呢?根据文献[42]所提出的方法思想:首先,由于|L|>|A|,由于日志中重复出现了活动 t,a_1,a_2,a_3,a_4,s 这些活动,因此我们可以对这些活动进行并行化的划分,这里可以选择活动 t,活动 a_1(a_2,a_3,a_4 和活动 a_1 在循环实例划分方面的功能等价)以及活动 s 作为循环标识进行讨论.

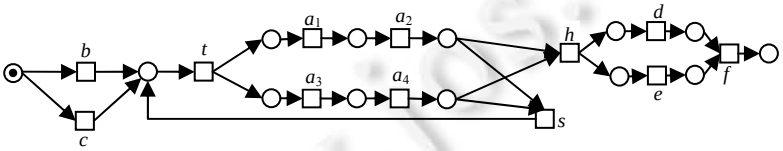


Fig.8 Incomplete loop example

图 8 非完全循环情况示例

当以活动 t 为循环标识时,单触发序列σ将被划分为$\sigma(t)$={⟨b⟩,⟨t,a_1,a_2,a_3,a_4,s⟩,…,⟨t,a_3,a_4,a_1,a_2,h,d,e,f⟩};若以活动 a_1 为循环标识时,单触发序列σ将被划分为$\sigma(a_1)$={⟨b,t⟩,⟨a_1,a_2,a_3,a_4,s,t⟩,…,⟨a_3,a_4,a_1,a_2,h,d,e,f⟩};若以 s 为循环标识时,单触发序列σ将被划分为$\sigma(s)$={⟨b,t,a_1,a_2,a_3,a_4⟩,⟨s,t…⟩,…,⟨s,t,a_3,a_4,a_1,a_2,h,d,e,f⟩}.根据这 3 种划分情况,利用过程挖掘算法(由于不存在噪声,以α算法为例)挖掘的结果分别如图 9(a)~图 9(c)所示.

© 中国科学院软件研究所 http://www.jos.org.cn

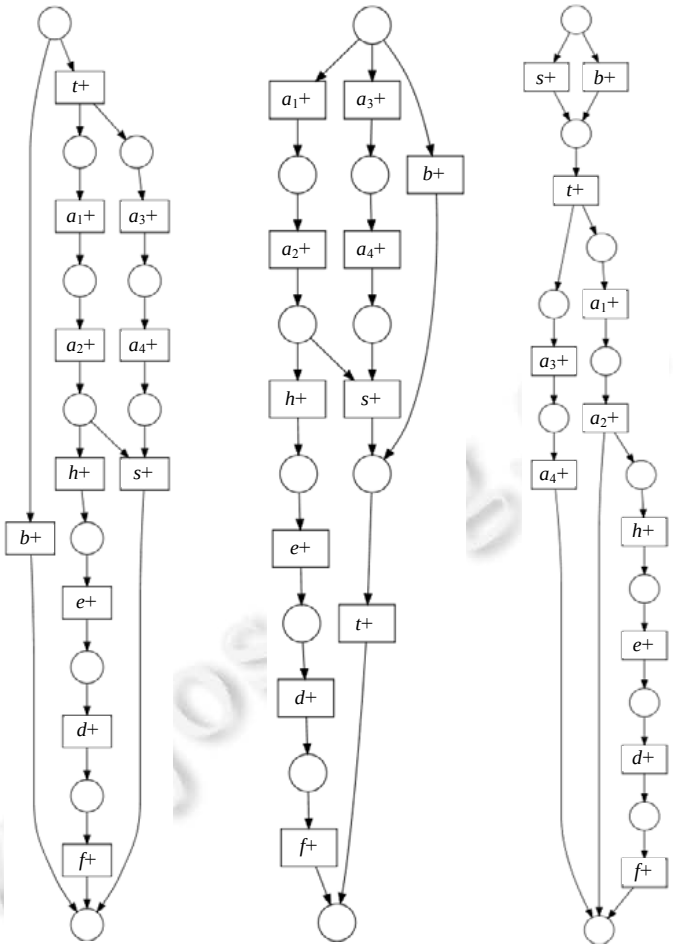


Fig.9 σ mining results using α algorithm

图 9 σ使用α算法挖掘结果

显然,虽然活动 a_1,a_2,a_3,a_4 均能够被正确地划分到相应的案例中,但是活动 b 和活动 h,d,e,f 的存在,破坏了循环与循环外部之间的关系,这就要求对于非完全循环的情况下必须首先判断哪些活动处于循环内、哪些活动不处于循环内,对循环外部的活动进行过滤,然后再针对剩下处于循环内的活动采用循环实例划分的方法.为判断活动是否处于某一循环内部,给出定义 22、定义 23.

定义 22(发生先于). 设活动 a,b 为活动集 A 中的两个活动,若存在一个步序列使得$[a\rangle\ldots[b\rangle$,则称活动 a 发生先于活动 b,记为 $a\mapsto b$,否则记为 $a\not\mapsto b$;设活动集 A,B,若对于$\forall a\in A,\forall b\in B$ 均有 $a\mapsto b$,则记为 $A\mapsto B$.

定义 23(循环归属). 设单触发序列对应的事件志记为 L,活动集为 A,且活动 $a\in A$ 为单触发序列 L 上的循环标识,则活动 $b\in A$ 与活动 a 同属于一个循环,当且仅当:

(1) 事件日志即 L 是完备的;

(2) $a\mapsto b\wedge b\mapsto a$.

比如本节中 $L=[\langle b,t,a_1,a_2,a_3,a_4,s,\ldots,t,a_3,a_4,a_1,a_2,s,d,e,f\rangle]$为完备的日志,对于活动 b 并不在循环内,因此对于循环标识(不妨设为 t)而言,$b\mapsto t$ 但是 $t\not\mapsto b$.因此,活动 b 不在循环内.同理可以判定活动 d,e,f,这样,我们可以在对循环进行处理时,将活动 b 和活动 d,e,f 首先排除掉,进而剩下的结构为一个完全循环情况.

© 中国科学院软件研究所 http://www.jos.org.cn

5.3 划分的日志完备性

下面来对判断活动是否在循环中的日志完备性进行讨论,因为只有完备的日志才能正确地发现活动是否在循环中,因此定义正确判断的日志完备性定义如下:

定义 24(日志完备性). 设 $p=(C,A;F,M_0)$为一个软件过程,其中包含循环 $LN=(C_L,A_L;F_L,M_e)$,L 为 p 上的过程日志当且仅当 $L\in\mathcal{P}(A^*)$且$\forall\sigma\in L$ 均为 p 上的一个点火序列,该序列以情态 M_i 开始,结束于情态 $M_o(M_i[\sigma\rangle M_o)$,且 $M_i,M_o\in R(M_0)$.L 为 p 上的能够正确判断活动是否归属于循环的完全过程日志当且仅当:

(1) 对于 p 上任意过程日志 L',均有$\rangle_{L'}\subseteq\rangle_L$;

(2) $\forall a\in A_L$,均有 a 出现在σ中;

(3) 设循环标识为 t,$\forall a\in A_L\wedge a\neq t$,则在$\sigma$中均有 $a\mapsto t$.

能够正确判断活动是否处于循环的完全过程日志的前两个条件同传统事件日志的完备性类似,主要是指活动间的依赖关系必须能够体现,同时活动也要在轨迹中出现过,因此对于单触发序列而言,循环外部如果出现了选择结构,而在该单触发序列其他地方没有出现过的话,这种选择结构是无法进行挖掘的.比如图 8 中的活动 c 由于没有在日志中出现,因此是不可能被挖掘出的.由于前两条同传统事件日志的完备性类似,此处不再赘述,可以参考文献[53]中对事件日志完备性定义.对于第(3)条要求,实质上是在说只要在循环中的出现过的活动在轨迹中至少要在循环标识之前出现过至少一次,比如图 10 中设循环标识为 t,在循环结束时存在一个活动 a 和活动 b 的选择关系,这样,如果在前边所有的循环实例中都只发生活动 a,而在循环结束时发生活动 b,比如形成轨迹$\langle\ldots,t,\ldots,a,\ldots,t,\ldots,a,\ldots,t,\ldots,b,c\rangle$,这样,通过循环标识 t 进行划分,只有最后的轨迹中包含活动 b,因此只存在 $t\mapsto b$ 但是 $t\not\mapsto b$,根据定义 23,活动 b 就不能在 t 所标识的循环中,这样的轨迹所形成的日志不是完备的,因此对于定义 24 中的第(3)个要求,在循环中的活动必须要在循环标识之前发生一次.

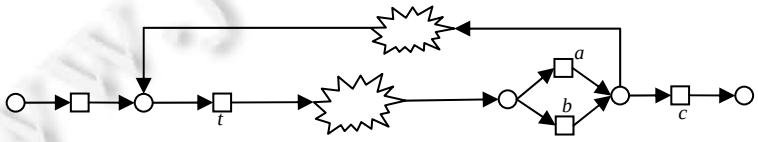


Fig.10 An example of a log completeness that correctly determines if an activity is in a loop

图 10 正确判断活动是否在循环中的日志完备性示例

实际应用中,可能受噪声的干扰,会导致某些情况对活动是否在循环中的情况造成干扰.为了避免这种情况的发生,提出启发式的判断方法来对噪声进行处理.

定义 25(循环归属度量). 设轨迹σ为定义在活动集 A 上的单触发序列,活动 a 为循环标识,记轨迹σ中序列“$a\ldots b\ldots a$”的频度为$|a\infty_\sigma b|=|\{1\leqslant i<j<k\leqslant|\sigma|,|\sigma(i)=a\wedge\sigma(j)=b\wedge\sigma(k)=a\wedge\sigma(t)_{\forall t\in(i,k)}\neq a\wedge\sigma(t)_{\forall t\in(i,j)}\neq b\}$.称

$$a\propto_\sigma b=\frac{|a\infty_\sigma b|+|b\infty_\sigma a|}{|a\infty_\sigma b|+|b\infty_\sigma a|+1}$$

为轨迹σ中活动 b 与循环标识 a 的循环归属值可记为 $C_\sigma^b(a)$.

根据定义 25,$a\infty_\sigma b$ 说明了活动 b 出现在活动两个重复的活动 a 之间,仅仅这一个条件仍无法确定二者处于一个循环中,因为可能二者处于并发活动中.因为若活动 b 和活动 a 处于同一循环的话,根据定义 23,则有 $a\mapsto b\wedge b\mapsto a$,这就要求还需要对 $b\infty_\sigma a$ 进行度量.一个无噪声的日志中,$|a\infty_\sigma b|$和$|b\infty_\sigma a|$是相等的,也就是说在度量时,二者在数量上应该相差不大,所以$|a\infty_\sigma b|$和$|b\infty_\sigma a|$之间的值应该共同增长,即,通过定义 25 进行度量.需要注意的是,条件$\sigma(t)_{\forall t\in(i,k)}\neq a\wedge\sigma(t)_{\forall t\in(i,j)}\neq b$ 说明,当两个重复的活动 a 之间可能出现多个活动 b 时,选择第 1 次出现的活动 b 进行计算.

根据设定的归属阈值来判断哪些活动和循环标识归属于同一循环.对于非循环中的活动,是无法完全挖掘的(比如选择结构);而对于循环中的活动,在日志完备的情况下,是可以通过划分的多个循环实例进行正确的挖

© 中国科学院软件研究所 http://www.jos.org.cn

掘,再将模型进行整合,最终完成整个单触发序列挖掘任务.

6 实验及案例分析

6.1 活动层挖掘

本节将通过两个实际的软件开发数据对本文第 4 节活动层提出的活动发现方法进行验证.

6.1.1 实验数据

两个实验数据集:开源软件 jEdit(http://www.jedit.org/)以及 ArgoUML(http://argouml.stage.tigris.org/)的开发过程日志.jEdit 和 ArgoUML 两个开源软件的代码均使用软件版本控制软件 SVN 进行维护,jEdit 开发日志能够通过 VisualSVN Server 的控制台使用 svn log 命令进行抽取,ArgoUML 开发日志抽取方法类似,不再赘述.

将 jEdit 对应的过程数据记为 JD 数据,将 ArgoUML 对应的过程数据记为 AD 数据.均使用 2010 年至今所提交的数据,通过对数据中的唯一标识 Revision 进行统计可知,JD 过程日志共有 1 995 个事件,AD 过程日志共 1 940 个事件.

根据第 4 节的相关算法对每个过程日志中的事件向量化后,JD 过程日志共产生 3 462 个特征词,AD 过程日志共产生 3 998 个特征词.本实验在特征抽取阶段,设置属性 *action*,*path* 以及 *msg* 的权值分别为 0.2,0.5,0.3.设置 *path* 的权值比 *msg* 高主要是考虑 *path* 相对较为规范,降低 *msg* 信息关联性较低对聚类结果的影响.

实验的运行平台为 Intel dual-core i5-3230M CPU(2.6GHz)且具有 12G RAM 的 PC 上.

6.1.2 实验步骤

实验设计方案如下:

(1) 将软件开发过程数据转化为过程日志;

(2) 对每个过程日志中的事件进行特征词抽取;

(3) 根据 WLVSM 将事件向量化,过程日志被转化为特征矩阵;

(4) 对特征矩阵进行模糊 FCM 聚类;

(5) 根据平均活动熵确定聚类结果,发现活动;

(6) 将事件与活动进行关联.

6.1.3 实验结果分析

根据本文第 4 节所提出的方法,实验结果如图 11~图 13 所示.图 11 为 AD 过程日志与 JD 过程日志利用模糊 FCM 聚类计算聚类数为 2~100 时平均活动熵的变化.聚类数太大或者太小均没有意义,如果聚类太多,也就是形成的活动太多,说明日志中每个信息间的相关性太小;如果聚类的个数太少,也就使得挖掘失去意义.本文中,两个数据集规模大致一样,因此此处考虑取 2~100 进行考察.

实验结果表明:JD 过程日志在聚类数为 10 的时候平均活动熵达到最小值为 0.88,而 AD 过程日志在聚类数为 31 的时候平均活动熵达到最小值为 0.70.故根据活动熵的定义,JD 过程日志应该聚类为 10 个活动,AD 过程日志应该聚类为 31 个活动.同时,根据图 11 还可知,AD 过程日志平均活动熵值比 JD 过程日志要低.说明了 AD 过程日志聚类的效果更加明显,事件的认真程度比 JD 过程日志要好.通过人为分析二者的 SVN 日志,同样发现 ArgoUML 的日志更加规范,结果符合实验预期.图 12 为 JD 过程日志与 AD 过程日志分别聚类为 10 个活动和 31 个活动时,每个活动包含的事件的分布图,其中,对 JD 过程日志而言,活动 6 包含的事件数目最多达到 758 个,也就是说在 JD 过程日志的 1 995 个事件,其中 758 个事件与活动 6 关联;同理,对于 AD 过程日志而言,第 13 个活动中包含事件数目最多,达到 208 个.图 13 为随着聚类数目的增加,整体算法的时间消耗.从图中可以看出,随聚类数目的增加,算法时间消耗基本符合一条直线;同时可以看出,ArgoUML 的时间消耗普遍比 jEdit 多.这是因为 ArgoUML 所抽取的特征词数量要比 jEdit 多,也就是特征词的数量会对算法的效率产生影响.

© 中国科学院软件研究所 http://www.jos.org.cn

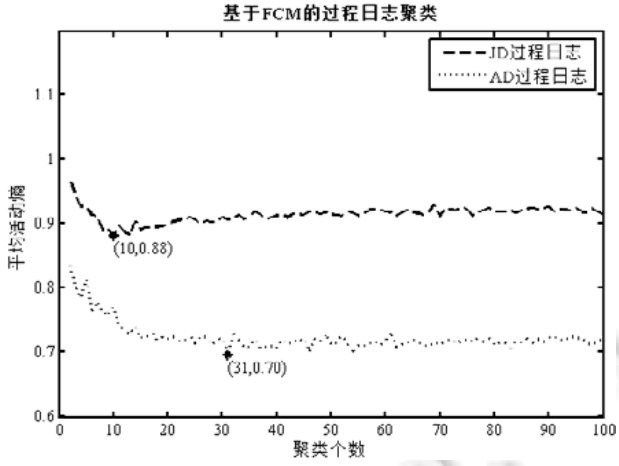


Fig.11 Fuzzy clustering of jEdit and ArgoUML using FCM

图 11 使用 FCM 对 jEdit 和 ArgoUML 模糊聚类

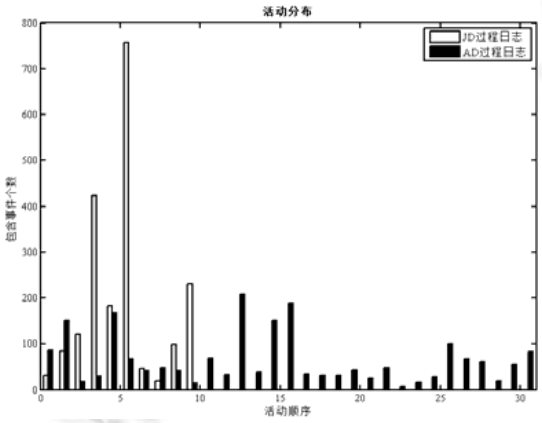


Fig.12 jEdit and ArgoUML mining activity distribution

图 12 jEdit 与 ArgoUML 的挖掘出活动分布情况

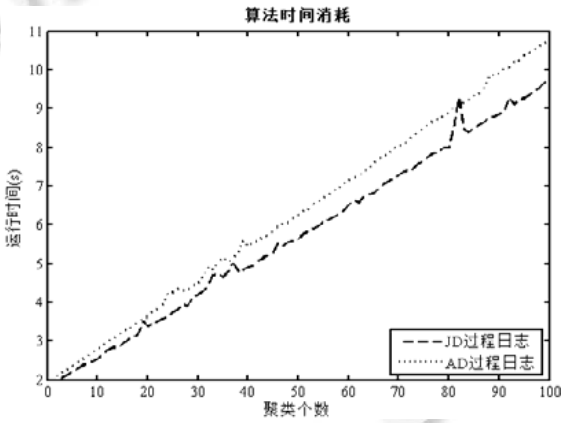


Fig.13 Time consumption of the algorithm

图 13 算法的时间消耗

在活动抽取完毕后,每个事件能够关联到一个活动上.比如能够对 SVN 日志的 XML 文件中的每个 logentry 元素添加一个 activity 标签以存储活动信息(如图 14 所示),“23823”的 logentry 对应于活动 A_6,故在 SVN 日志中添加活动标签及值⟨activity⟩A_6⟨/activity⟩;同理,“23824”的 Logentry 元素添加活动标签及值⟨activity⟩A_{10}⟨/activity⟩.发现事件对应的活动信息后,就能够使用过程挖掘的相关方法.

与 Rubin 的方法[7]进行对比:Rubin 的方法主要是基于 Mapping 来完成的,首先定义若干活动,比如图 15 中定义的 4 种活动——DES(设计),CODE(编码),TEST(测试),REV(复审);其次,根据所涉及的文档的名字与上述 4 种活动进行映射;最后,根据映射的活动来进行过程挖掘.因此,基于 Mapping 的方法主要是通过对路径中的关键字进行过滤来完成活动的映射,这种方法的优势在于速度较快,但存在的问题是不够灵活、映射结果不准确、活动发现的粒度较粗、挖掘出的过程模型也只可能包含若干预先定义好的活动上.这样挖掘出来的过程模型的粒度自然也比较粗,使得模型缺乏指导意义.通过本文挖掘出来的活动信息根据时间的先后顺序,SVN 日志信息可以描述为 L=[⟨a_1,a_3,a_5,a_5,a_4,a_2,a_6,a_4,a_3,a_1,a_4,a_5,a_5,a_5,a_6,a_2,a_4,a_3,a_4,a_1,a_6,a_5,a_4,a_2,a_1,a_5,a_3,a_2,a_4,a_6,a_4,a_3,a_4,a_6,a_1,a_4,a_5,a_2,a_1,a_2,a_3,a_4,s,t,…⟩]的单触发序列,这样就非常有助于下一步过程层的挖掘.

© 中国科学院软件研究所 http://www.jos.org.cn

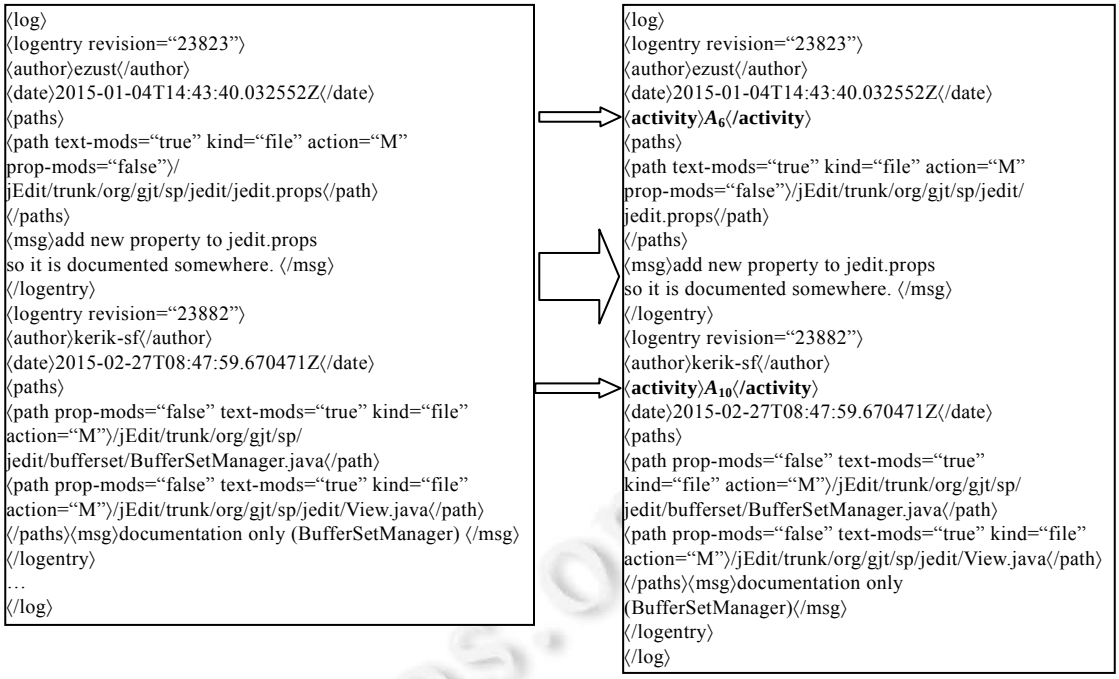


Fig.14 Insert activity information into the SVN log

图 14 插入活动信息到 SVN 日志中

Document Log

Document	Date	Author
project1/models/design.mdl	01.01.05 14:30	designer
project1/src/Code.java	01.01.05 15:00	developer
project1/tests/testPlan.xml	05.01.05 10:00	qaengineer
project1/docs/review.pdf	07.01.05 11:00	manager
project2/models/design.mdl	01.02.05 11:00	designer
project2/tests/testPlan.xml	15.02.05 17:00	qaengineer
project2/src/NewCode.java	20.02.05 09:00	developer
project2/docs/review.pdf	28.02.05 18:45	designer
project3/models/design.mdl	01.03.05 11:00	designer
project3/models/verification.xml	15.03.05 17:00	qaengineer
project3/src/GenCode.java	20.03.05 09:00	designer
project3/review/Areview.pdf	28.03.05 18:45	manager

Filtered Log

Document
DES
CODE
TEST
REV
DES
TEST
CODE
REV
DES
VER
CODE
REV

Fig.15 Activity discovery result based on mapping method[7]

图 15 基于 mapping 方法的活动发现结果[7]

6.2 过程层挖掘

本节将对本文第 5 节的过程层挖掘方法进行验证.首先,使用具有典型特征(包含顺序、并发、迭代、非自由选择等结构)的过程模型产生的数据对算法的正确性、合理性以及性能等方面进行验证;其次,限于篇幅原因,仅使用过程层挖掘中提到的循环实例划分以及文献[54]中提到的挖掘方法对第 6.1 节的中提到的 jEdit 的过程日志进行过程模型的发现.

6.2.1 实验数据

采用的数据分别为:

1) 由后文图 18 的过程模型生成长度为 2 000 的单触发序列(简称 CG).
2) 由后文图 18 的过程模型生成从开始执行到结束的长度为 1 996 的单触发序列(简称 FW),其中,活动 b 出现 1 次,t,a_1,a_2,a_3,a_4 出现 332 次,s 出现 331 次,h,d,e,f 出现 1 次.
3) 由图 18 包含嵌套循环、非完全循环情况、并发与循环交织过程模型生成从开始执行到结束的长度

© 中国科学院软件研究所 http://www.jos.org.cn

为 3 000 的单触发序列,同时,随机混入 50 个噪声(简称 NG),其中,活动 b 出现 1 次,t,a_4 出现 107 次,a_1,a_2,a_6 出现 535 次,a_5 出现 428 次,a_3 出现 642 次,s 出现 106 次,h,d,e,f 出现 1 次.

4) 给定包含 50 个活动集的随机生成长度为 2 000、包含 200 个案例(该案例仅用来对划分后的结果正确性进行验证)的单触发序列(简记 RD).

CG、FW 以及 NG 数据本身具有参考模型,每个事件均能够确定其所属的循环实例.CG 数据主要是为了验证模型完全是循环时的发现结果(如图 16 所示),FW 数据主要是为了验证模型是非完全循环的情况下划分的正确率,NG 数据主要是为了验证模型同时包含嵌套循环、非完全循环情况、并发与循环交织时的循环划分的正确率(如图 17 所示),RD 数据是为了验证随机给定事件及活动的划分正确率.

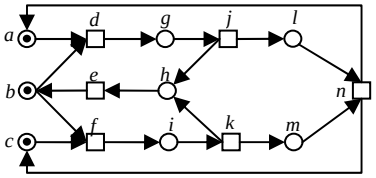


Fig.16 Complete loop situation

图 16 完全循环情况

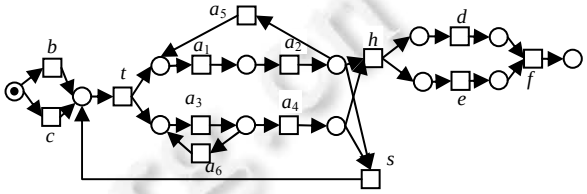


Fig.17 Nested loop situation

图 17 嵌套循环情况

6.2.2 实验步骤

实验设计方案如下.

(1) 对 CG、FW 以及 NG 数据中每个事件的循环实例进行标注,用于检验划分结果.

(2) 将每组数据作为输入,循环标识每组数据分别选择 3 个:重复活动中第 1 个出现的、重复活动中最后一个出现的以及重复次数最多的.

(3) 根据所选择的循环标识对每组数据中的每个活动的循环归属值进行计算,比如活动 b 相对于循环标志 a 的循环归属值为 $C_\sigma^b(a)$,循环归属阈值记为 δ,若 $C_\sigma^b(a)<\delta$,则活动 b 属于循环标志 a 所对应的循环;否则不属于.

(4) 根据循环归属值,对不属于循环的活动,从日志中进行过滤,仅保留这些活动与紧邻活动的关系.

(5) 对过滤后的日志进行循环实例划分.

(6) 对划分结果进行评估,评估标准为“划分正确率=正确划分的循环实例数/总循环实例数”.

(7) 统计划分时间.

(8) 将划分后的多条循环实例作为案例,选择某种挖掘算法进行挖掘.

(9) 将已过滤的活动(顺序结构)与挖掘出的模型进行拼接,从而得到最终的挖掘结果.

6.2.3 实验结果分析

根据实验方法,最终得到的实验结果如图 18~图 27 所示.

CG 数据每个活动相对于循环标识 d,e,f 的循环归属值如图 18 所示,其中,循环标识 d 相对于其自身的循环归属无需计算(此处设为 0),因此在横坐标 d 处只有两个值(这两个值分别表示活动 d 相对于循环标识 e 的循环归属值与活动 d 相对于循环标识 f 的循环归属值);同理,横坐标为 e,f 处也只有两个值.通过图 18 可以发现,无论选择哪个循环标识,均能够正确地判断出哪些活动应该归属于循环标识所在的循环.比如选择循环标识 d 时,其他活动相对于循环标识的循环归属值几乎相同并都接近 1,因此可以得出所有的活动都处于同一个循环中.同时,所有活动相对于循环标识 e 的循环归属值均较高,最终可选择循环标识 e 进行案例划分,因此,CG 数据能够正确地进行归属.

FW 数据每个活动相对于循环标识 t,a_1,s 的循环归属值如图 19 所示,因为 3 个循环归属值的结果较为接近,则以循环标识 s 进行举例.其中,b,h,e,d,f 相对于循环标识 s 的循环归属值均为 0,则活动 b,h,e,d,f 不属于循环标识 s 所在的循环,通过模型进行验证发现,这些活动 b,h,e,d,f 的确是处于循环之外的.

© 中国科学院软件研究所 http://www.jos.org.cn

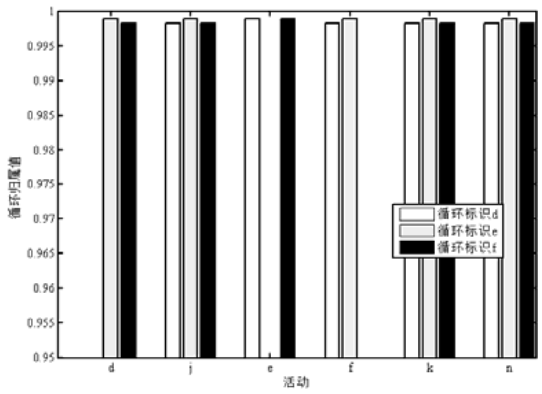


Fig.18 CG data loop ownership value

图 18 CG 数据循环归属值

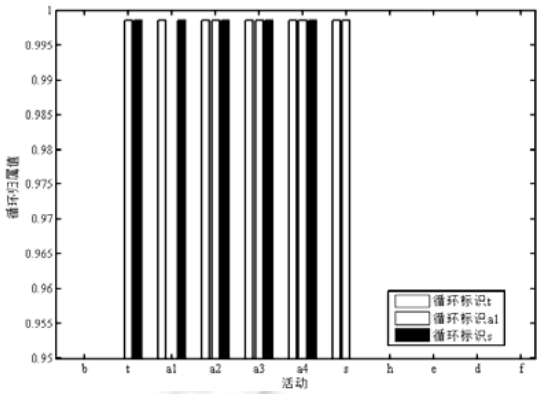


Fig.19 FW data loop ownership value

图 19 FW 数据循环归属值

NG 数据每个活动相对于循环标识 t,a_3,s 的循环归属值如图 20 所示,NG 数据随机混入了 50 个噪声,这些噪声可能与活动名称相同,也可能不同,用以验证本文提出的循环归属的度量方法.通过图 20 可以发现,由于噪声的存在,使得各个循环归属值参差不齐,但是无论对于哪个循环标识,活动 b,h,e,d,f 都显然不属于任一个循环标识所在的循环中.若以活动 t 为循环标识且阈值为 0.9 时,则 a_1~a_6 以及活动 s 均应处于活动 t 所在的循环中,阈值的设定需要人为地对该参数进行调节,通过整个数据各个活动对应的循环归属值来确定.NG 数据同时也是由一个嵌套循环所产生的,对于该类情况,事实上,每个嵌套循环总会有一个最外围的循环,对于单触发序列而言,只需对最外围的循环进行循环实例划分即可,内部嵌套的循环通过传统挖掘算法对划分后的多条循环实例进行挖掘是能够发现的,且内部嵌套的循环对于外围循环的循环归属计算不会产生影响.这是因为循环归属的度量对于内部重复的活动只会计算一次,比如,活动 a_3 和 a_6 是一个内部循环,但无论内部循环重复多少次,这些发生次数总会处于两个循环标识 t 之间,同时,这些重复的 a_3 与 a_6 都只会计算一次.另外,为了进一步验证循环归属度量的正确性,实验分别计算了以活动 a_1~a_6 以及 s,t 为循环标识时,每个活动对应的精确度和召回率,结果如图 21 所示.可以看出,几乎所有活动作为循环标识时,通过循环归属度量方法都能有一个较好的归属判定结果.

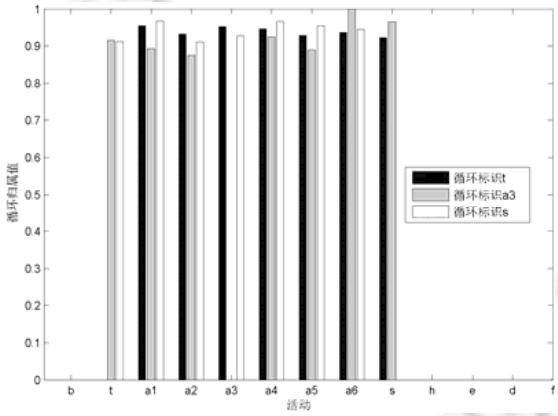


Fig.20 NG data loop ownership value

图 20 NG 数据循环归属值

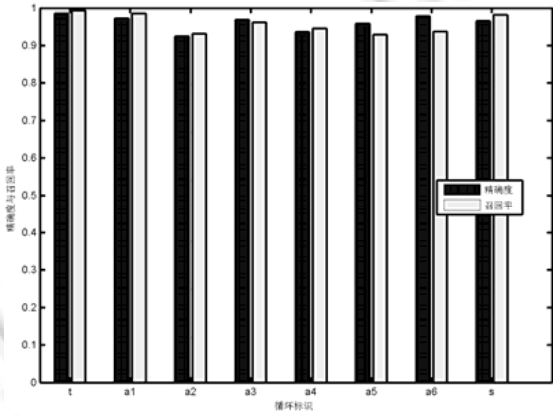


Fig.21 Accuracy and recall rate corresponding NG data loop identifier

图 21 NG 数据循环标识对应的精确度与召回率

RD 数据循环归属值如图 22 所示,因为每个活动出现的概率基本相同,因此所有活动被归属到同一个循环中,计算结果符合预期.

© 中国科学院软件研究所 http://www.jos.org.cn

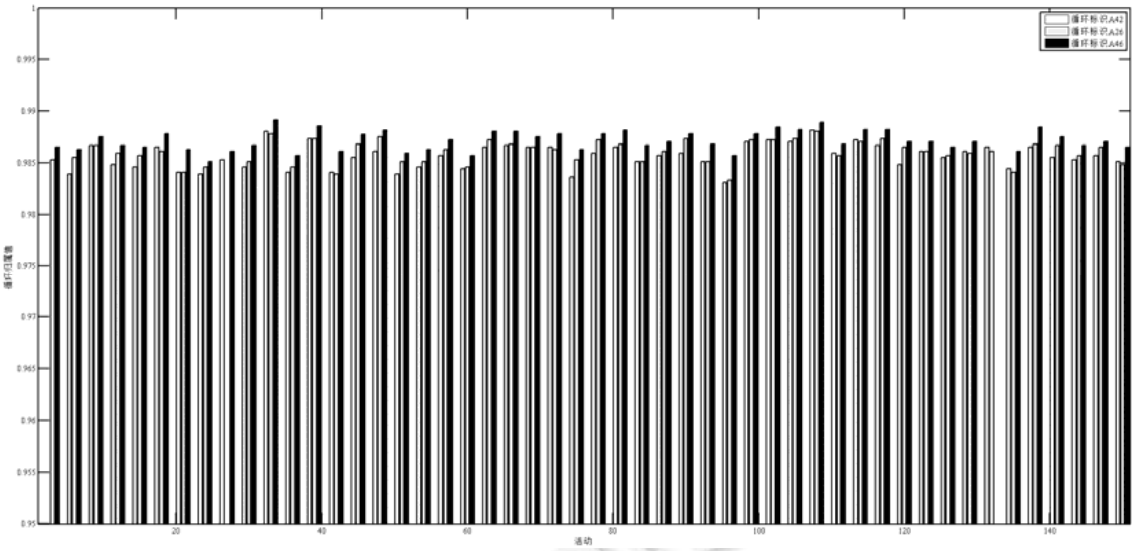


Fig.22 RD data loop ownership value

图 22 RD 数据循环归属值

然后,对发现的循环进行循环实例划分,划分的正确率度量结果如图 23 所示,CG,FW 以及 NG 都能够完全正确划分符合预期.正确划分的主要原因是这些数据集能够有一个极好的循环归属度量,从而选择循环标识进行划分时得到较好的划分结果.数据 RD 每次执行产生的随机数不相同,因此为了保证正确性,其划分正确率是执行 50 次取得平均值,但是正确率在 3 组数据中仍然最低.经分析发现,这是因为随机数的产生器产生随机数的概率是相同的,即,下一个活动出现的几率相同,这样产生的序列中所有的活动都被判断属于了同一个并发活动集,故划分得到案例与随机产生的案例之间偏差较大,但是这也是和预期相符的.图 24 为 4 组数据对应循环实例划分平均时间,RD 数据量较大,FW 数据需要先对循环进行判断,因此划分时间稍高于其他两组.

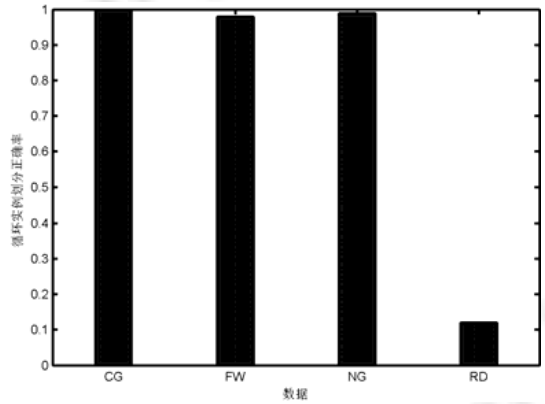


Fig.23 Correct division rate corresponding 4 groups of data

图 23 4 组数据对应划分正确率

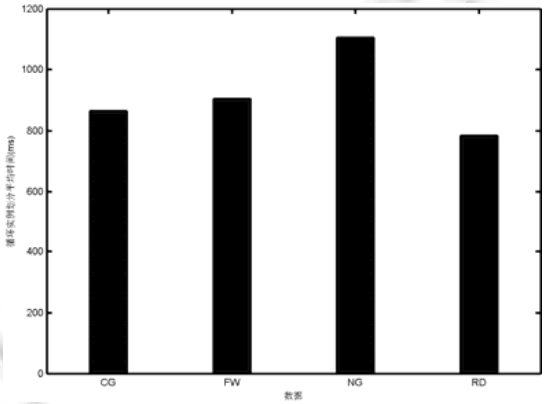


Fig.24 Average time corresponding 4 groups of data

图 24 4 组数据对应划分平均时间

最后,通过对前 3 组数据对应的单触发序列进行循环实例划分,划分后再利用任意过程挖掘算法进行挖掘(本实验使用的是启发式挖掘方法),挖掘结果如图 25~图 27 所示.可以看出,无论是哪组数据的挖掘结果,与产生日志所使用的过程模型都基本一致,不会再产生本节开头所论述的直接使用过程挖掘算法而导致开头挖掘出

© 中国科学院软件研究所 http://www.jos.org.cn

错的情况.需要说明的是,图 26 与图 27 中活动 *c* 处于循环外部且处于选择结构中,所产生的日志中不包含任何信息,因此这是无法发现的;另外,模型尾部活动 *h*,*e*,*d*,*f* 因为在日志中只出现 1 次,因此会被挖掘成一个顺序结构,这些都符合过程挖掘的日志完备性假设.

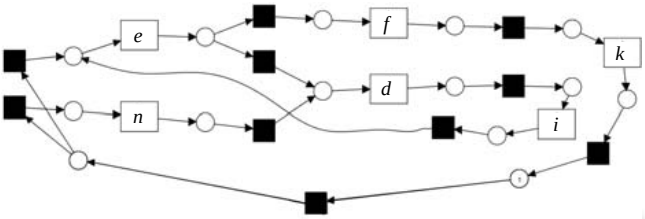


Fig.25 Mined process model according to activity *n* as loop identifier based on CG data

图 25 数据 CG 以活动 *n* 为循环标识挖掘出的过程模型

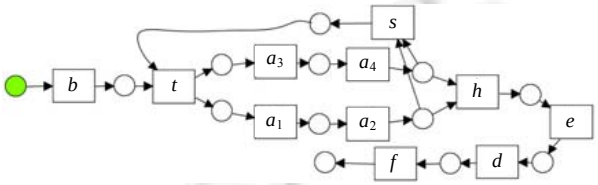


Fig.26 Mined process model according to activity t as loop identifier based on FW data

图 26 数据 FW 以活动 *t* 为循环标识挖掘出的过程模型

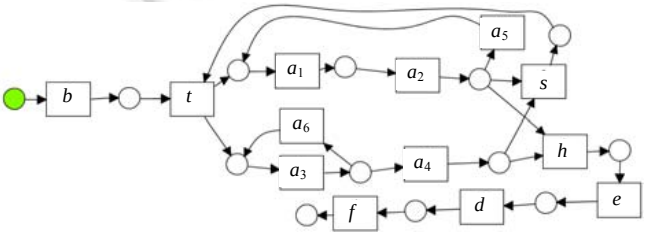


Fig.27 Mined process model according to activity *t* as loop identifier based on NG data

图 27 数据 NG 以活动 *t* 为循环标识挖掘出的过程模型

6.2.4 基于 JD 数据的过程模型发现

限于篇幅原因,本节仅对开源软件 jEdit 的开发过程日志进行挖掘,ArgoUML 过程日志的挖掘是类似的.

如前文所述,JD 数据集为开源文本编辑软件 jEdit 的 SVN 从 2010 年至今所提交(check in)的日志数据,共有 1 995 条 Logentry 元素,故对应 1 995 个事件,根据第 4 节的研究发现,在聚类数为 10 的时候,平均活动熵达到最小值.因此该数据集被聚类为 10 个活动,通过将活动信息反写入日志文件中,这样就形成一个典型的单触发序列循环实例划分问题.通过使用第 5 节的过程层挖掘相关研究方法,对该单触发序列进行过程层挖掘,即进行循环实例的划分.为了进行软件过程挖掘,我们同时开发了原型系统 SPMining,利用该系统划分后的案例数目为 91 个,活动 J_3 被作为循环标识来进行循环实例的划分.

将事件日志导入到 SPMining 中来对过程信息进行统计(如图 28 所示),平均每个案例中包含的事件信息为 22 个,最短的案例包含两个事件,最多的案例包含 74 个事件.

过程层挖掘的目的是能够找到事件的案例信息,以此来适配软件过程的挖掘,再根据过程挖掘方法对 JD 数据集进行过程模型的挖掘,挖掘结果如图 29 所示.

© 中国科学院软件研究所 http://www.jos.org.cn

Fig.28 Result of JD data loop instance divided by prototype system SPMining

图 28 原型系统 SPMining 对 JD 数据循环实例划分结果

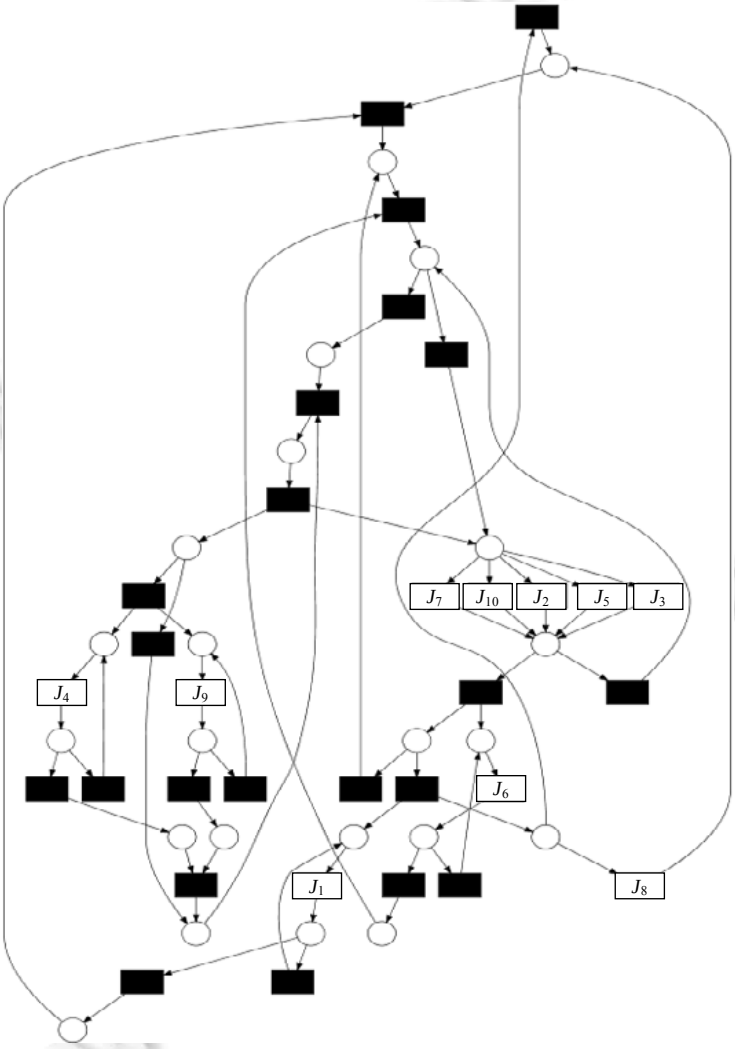


Fig.29 Software process model mined from JD data set

图 29 JD 数据集挖掘出的软件过程模型

© 中国科学院软件研究所 http://www.jos.org.cn

通过图 29 可知,具有显著特征的是活动 J_7,J_{10},J_2,J_5,J_3 处于一个选择结构中,J_6,J_4,J_9,J_1 均处于一个短循环中,日志中 J_6,J_4 两个活动出现次数最多,也证实了该活动处于短循环结构中.挖掘出 JD 数据集对应的过程模型后,每个活动对应的SVN日志应该被存储于活动信息内部并能够被查看,比如当点击活动 J_1 时,绑定在 J_1 中的SVN事件 revision 号对应关系见表 3.

Table 3 SVN log revision number bound to activity J_1

表 3 绑定在活动 J_1 上的 SVN 日志 revision 号

活动	J_1	J_1	J_1	J_1	J_1	J_1	J_1	J_1	J_1	J_1	J_1	J_1	J_1	J_1	J_1	J_1	J_1
revision	23820	23335	23298	23166	23165	23017	23011	22935	22931	22919	22833	22812	22498	22261	21581	20638	20635
活动	J_1	J_1	J_1	J_1	J_1	J_1	J_1	J_1	J_1	J_1	J_1	J_1	J_1	J_1	J_1	J_1	J_1
revision	20470	20428	20212	20132	20130	20126	20125	20124	19975	19552	19551	19240	19211	19193	17727	20470	20428

举例而言,SVN 日志中 revision 号为 23820 以及 revision 号为 23298 的记录如图 30 所示,通过分析可以发现,二者均对 FAQ 信息进行了修改.

```
⟨logentry revision="23820"⟩
⟨author⟩ezust⟨/author⟩
⟨date⟩2015-01-04T01:22:00.224072Z⟨/date⟩
⟨paths⟩
⟨path text-mods="true" kind="file" action="M"
prop-mods="false"⟩/jEdit/trunk/doc/FAQ/faq-install.xml⟨/path⟩
⟨path text-mods="true" kind="file" action="M"
prop-mods="false"⟩/jEdit/trunk/doc/FAQ/faq-use.xml⟨/path⟩
⟨/paths⟩
⟨msg⟩FAQ: Reformatted whitespace and made small updates. ⟨/msg⟩
⟨/logentry⟩
```

```
⟨logentry revision="23298"⟩
⟨author⟩ezust⟨/author⟩
⟨date⟩2013-10-29T15:41:30.563399Z⟨/date⟩
⟨paths⟩
⟨path prop-mods="false" text-mods="true" kind="file"
action="M"⟩/jEdit/trunk/doc/FAQ/faq-use.xml⟨/path⟩
⟨paths⟩
⟨msg⟩How to toggle auto-indent FAQ. ⟨/msg⟩
⟨/logentry⟩
```

Fig.30 Mining result of JD data

图 30 JD 组数据活动层发现结果图示

7 相关工作

过程挖掘作为数据挖掘与业务过程管理之间的桥梁,已经成为了现代组织用于管理复杂运作流程的一种重要工具[9].大量现存的过程挖掘算法已经对过程挖掘进行了深入的研究,但是几乎所有方法都是以事件日志的案例信息必不可缺为前提的.

然而,软件过程与传统业务过程最大的区别在于单实例性,也就是过程实例过少,在此背景下,本文提出双层次的软件开发过程挖掘方法,通过从过程日志中发现缺失的活动信息以及案例信息来支持过程挖掘算法.

针对软件过程挖掘,文献[32]提出一种增量式挖掘方法,该方法的数据主要来自软件开发的版本管理系统的日志信息,这些信息包含有实施过程的控制流、文档以及资源信息.且该文献作者认为,这些信息同样也能够通过 ERP 或者 CRM 系统获取,因此也适用于这些系统的挖掘.但是由于软件开发版本所记录的日志信息往往是单一软件过程实例的执行记录,即只包含了一个过程实例,对于该问题,作者试图使用该软件不同开发语言产生的日志信息进行挖掘[7]来规避单触发序列的问题.增量式挖掘方法的原理是,由于初始案例较少,因此将挖掘出一个粒度较粗的过程模型,一旦有其他数据产生,就可以重新细化原来的过程模型.这样,随着日志数量的增加,过程模型的精细程度也就越来越高.可见,该方法无法有效地解决单触发序列问题.

文献[24,55]提出了一种遗传过程挖掘方法,遗传过程挖掘方法的原理和所有遗传算法一样,利用交叉和变异进行迭代,在交叉中使用两个父代个体去创造新的后代,下一步再使用变异修改得到子模型.如此迭代,直到找到一个达到期望适应度的模型.理论上,遗传过程挖掘可以通过日志发现任何过程模型,但事实上,其算法收敛时间过长,实践中常常不能出现一个达到期望适应度的模型,故常常添加结束标准(比如,连续 10 代无法产生更好个体时结束).因此,尽管遗传过程挖掘在活动数目较少时能够发现单触发序列中的循环结构,但并不能很好地适应于包含大量活动的日志.

© 中国科学院软件研究所 http://www.jos.org.cn

区别于现有的软件过程挖掘方法,本文针对软件过程日志的单实例性特征,提出了双层次的挖掘方法.该方法首先从过程日志中通过聚类发现活动信息,然后以活动发生的时序顺序转化为单触发序列,进而通过循环将单触发序列划分为多个循环实例,这些循环实例能够作为案例信息支持传统过程挖掘方法的使用.

由于通过过程日志发现活动信息后将形成单触发序列,因此本文以单触发序列中的循环为着手点,将单触发序列进行循环实例划分,以达到发现循环部分的案例信息,进而用于过程挖掘.因此,本文方法是对当前过程挖掘方法的完善,在处理单触发序列方面,相对于其他算法具有明显优势.

对于单触发序列的研究可能还与序列挖掘、情节(episode)挖掘以及 Petri 网语言等方面相关,其中,序列挖掘的目的是为了发现序列中的频繁序列[56];情节挖掘[57]是为了发现频繁的情节,一段情节定义了一个偏序关系,其挖掘方法常常使用滑窗的方式进行,情节挖掘不支持发现并发活动,序列挖掘和情节挖掘都是关联规则学习的变体,它们都只考虑顺序关系而不支持发现并发、选择和循环;Petri 网语言[58]主要是指 Petri 网及其所产生语言之间的关系,它主要被用来分析 Petri 网的行为,模型对应语言,语言产生轨迹,通过该过程相对容易,但是从轨迹发现语言(或模型)相对较难.总之,上述方法都不能完全有效地解决单触发序列的挖掘问题,而对单触发序列的解决是本文双层次挖掘方法与其他软件过程挖掘方法最大的区别.

8 结束语

当前,软件过程建模问题已经成为限制软件过程研究的核心问题,过程挖掘作为数据科学的一种最佳实践,在多种领域中得到了应用和推广,以二者结合的方法来对软件过程中涉及到的数据和挖掘方法还鲜有文献进行讨论,因此,本文在此背景下提出了双层次的软件过程挖掘方法.该方法将分为活动层及过程层.

- 在活动层对过程日志中的特征词进行抽取;为了能够有效区别某些单词的重要程度,提出了 WSLVM 模型对每条记录的特征进行向量化,将事件转化为特征向量集,然后对特征向量集进行聚类,将聚类的结果作为活动与事件进行绑定;提出了利用模糊聚类方法并结合平均活动熵来对聚类结果进行确定的方法.
- 在过程层,基于启发式的单触发序列挖掘方法,针对非完全循环的情况下的事件日志的完备性问题进行了研究,提出了非完全循环情况下的循环归属条件.

通过两个真实的软件过程数据来分别进行实验,证明各层次方法的正确性及可行性.

未来工作中,我们拟考虑如下若干方面的工作.

(1) 针对当前的过程挖掘算法进行改进,提出适用于软件过程挖掘的针对性算法.软件过程挖掘的目标在于能够快速地发现一个简洁、合理、优质的过程模型以支持软件开发活动的进行.尽管当前已经存在了一些过程挖掘算法,但现实情况是,软件过程相对复杂,挖掘算法效率较低,挖掘结果不够简洁,不能在多种度量属性间获得平衡.

(2) 从数据来源来看,当前仍然没有一种面向过程挖掘与分析的软件过程仓库管理系统,当前软件过程中的数据都是被动式、面向开发者的.所谓被动式是指当前的执行数据是被动式进行存储的,而不是有目的地对事件信息进行存储.因此,有必要对面向过程的软件过程数据管理系统进行研究.

(3) 软件过程挖掘工具.当前,过程挖掘工具都仅限于客户端、专业人士,这极大地限制了过程挖掘方法的普及和推广.当前,过程挖掘方法已经较为成熟,能够利用现有的优秀的前端展示技术对结果进行展示,将过程挖掘的业务逻辑封装成服务以供不同需求对这些业务服务的使用.

References:

[1] CMMI PT. CMMI® for development, improving processes for better products. Version 1.2. Pittsburgh: Software Engineering Institute, 2006.

[2] Mordal K, Anquetil N, Laval J, Serebrenik A, Vasilescu B, Ducasse S. Software quality metrics aggregation in industry. Journal of Software Evolution & Process, 2013,25(10):1117−1135. [doi: 10.1002/smr.1558]

© 中国科学院软件研究所 http://www.jos.org.cn

[3] Lonchamp J. A structured conceptual and terminological framework for software process engineering. In: Proc. of the 2nd Int'l Conf. on Software Process, Continuous Software Process Improvement. Berlin: IEEE, 1993. 41−53. [doi: 10.1109/SPCON.1993.236823]

[4] Kaur R, Sengupta J. Software process models and analysis on failure of software development projects. Int'l Journal of Scientific and Engineering Research, 2011,2(2):1−4.

[5] Maciel RSP, Gomes RA, Magalhães AP, Silva BC, Queiroz JPB. Supporting model-driven development using a process-centered software engineering environment. Automated Software Engineering, 2013,20(3):427−461. [doi: 10.1007/s10515-013-0124-0]

[6] Kindler E, Rubin V, Schäfer W. Activity mining for discovering software process models. Software Engineering, 2006,79:175−180.

[7] Rubin V, Günther CW, van der Aalst WMP, Dongen EKBFV, Schafer W. Process mining framework for software processes. In: Wang Q, Pfahl D, Raffo DM, eds. Proc. of the Software Process Dynamics and Agility, Vol.4470. Berlin, Heidelberg: Springer-Verlag, 2007. 169−181. [doi: 10.1007/978-3-540-72426-1_15]

[8] Rubin V, Lomazova I, van der Aalst WMP. Agile development with software process mining. In: Proc. of the 2014 Int'l Conf. on Software and System Process. Nanjing: ACM Press, 2014. 70−74. [doi: 10.1145/2600821.2600842]

[9] van der Aalst WMP. Process Mining: Discovery, Conformance and Enhancement of Business Processes. Berlin, Heidelberg: Springer- erlag, 2011. [doi: 10.1007/978-3-642-19345-3]

[10] Hindle A, Godfrey MW, Holt RC. Software process recovery using recovered unified process views. In: Marinescu R, ed. Proc. of the 2010 IEEE Int'l Conf. on Software Maintenance (ICSM). IEEE, 2010. 1−10. [doi: 10.1109/ICSM.2010.5609670]

[11] Carmona J, Cortadella J. Process discovery algorithms using numerical abstract domains. IEEE Trans. on Knowledge & Data Engineering, 2014,26(12):3064−3076. [doi: 10.1109/TKDE.2013.156]

[12] Yu SS, Zhou SG, Guan JH. Software engineering data mining: A survey. Journal of Frontiers of Computer Science & Technology, 2012,6(1):1−31 (in Chinese with English abstract). [doi: 10.3778/j.issn.1673-9418.2012.01.001]

[13] Xie T, Thummalapenta S, Lo D, Liu C. Data mining for software engineering. Computer, 2009,42(42):55−62. [doi: 10.1109/MC.2009.256]

[14] Rodriguez D, Garcia E, Sanchez S, Nuzzi SRS. Defining software process model constraints with rules using OWL and SWRL. Int'l Journal of Software Engineering and Knowledge Engineering, 2010,20(4):533−548. [doi: 10.1142/S0218194010004876]

[15] Li T. An Approach to Modelling Software Evolution Processes. Springer-Verlag, 2008. [doi: 10.1007/978-3-540-79464-6]

[16] He KQ, Li B, Ma Y, Huang Y. The key technology of software engineering in big data age. Communications of the China Computer Federation, 2014,10(3):8−18 (in Chinese with English abstract).

[17] Wang QX, Mei H. Big data and software engineering. Communications of the China Computer Federation, 2014,10(3):6−7 (in Chinese with English abstract).

[18] Van der Aalst WMP, Adriansyah A, De Medeiros AKA, Arcieri F, Baier T, Blickle T, Bose JC, Brand PVD, Brandtjen R, Buijs J. Process mining manifesto. In: Daniel F, Barkaoui K, Dustdar S, eds. Proc. of the Lecture Notes in Business Information Processing. Berlin, Heidelberg: Springer-Verlag, 2011. 169−194. [doi: 10.1007/978-3-642-28108-2_19]

[19] Van der Aalst WMP. Extracting event data from databases to unleash process mining. In: Brocke JV, Schmiedel T, eds. Proc. of the BPM—Driving Innovation in a Digital World. Springer Int'l Publishing, 2015. 39−47. [doi: 10.1007/978-3-319-14430-6_8]

[20] Van der Aalst WMP, Weijters T, Maruster L. Workflow mining: Discovering process models from event logs. IEEE Trans. on Knowledge and Data Engineering, 2004,16(9):1128−1142. [doi: 10.1109/TKDE.2004.47]

[21] Weijters AJMM, van der Aalst WMP, De Medeiros AKA. Process mining with the heuristics miner-algorithm. BETA Working Paper Series 166, Eindhoven: BETA Publisher in Eindhoven University of Technology, 2006.

[22] Dongen BFV, Busi N, Pinna G, van der Aalst WMP. An iterative algorithm for applying the theory of regions in process mining. In: Reisig W, Hee KV, Siedlce KW, eds. Proc. of the Workshop on Formal Approaches to Business Processes and Web Services. Publishing House of University of Podlasie, 2007. 36−55.

[23] Bergenthum R, Desel J, Lorenz R, *et al*. Process mining based on regions of languages. In: Alonso A, Dadam P, Rosemann M, eds. Proc. of the Int'l Conf. on Business Process Management (BPM 2007). Berlin, Heidelberg: Springer-Verlag, 2007. 375−383. [doi: 10.1007/978-3-540-75183-0_27]

[24] De Medeiros AKA, Weijters AJMM, van der Aalst WMP. Genetic process mining: An experimental evaluation. Data Mining and Knowledge Discovery, 2007,14(2):245−304. [doi: 10.1007/s10618-006-0061-7]

[25] Günther CW, van der Aalst WMP. Fuzzy mining—Adaptive process simplification based on multi-perspective metrics. In: Alonso A, Dadam P, Rosemann M, eds. Proc. of the Int'l Conf. on Business Process Management (BPM 2007). Berlin, Heidelberg: Springer-Verlag, 2007. 328−343. [doi: 10.1007/978-3-540-75183-0_24]

© 中国科学院软件研究所 http://www.jos.org.cn

[26] Werf JMVD, Dongen BFV, Hurkens CAJ, Serebrenik A. Process discovery using integer linear programming. Fundamenta Informaticae, 2009,94(3-4):387−412. [doi: 10.3233/FI-2009-136]

[27] Cook JE, Wolf AL. Discovering models of software processes from event-based data. ACM Trans. on Software Engineering and Methodology (TOSEM), 1998,7(3):215−249. [doi: 10.1145/287000.287001]

[28] Rubin V, Günther CW, van der Aalst WMP, Kindler E, Dongen BFV, Schäfer W. Process mining framework for software processes. In: Wang Q, Pfahl D, Raffo DM, eds. Proc. of the Int'l Conf. on Software Process, Software Process Dynamics and Agility. Berlin, Heidelberg: Springer-Verlag, 2007. 169−181. [doi: 10.1007/978-3-540-72426-1_15]

[29] Garg PK, Bhansali S. Process programming by hindsight. In: Proc. of the 14th Int'l Conf. on Software Engineering. 1992. 280−293. [doi: 10.1145/143062.143128]

[30] Senin P. Software trajectory analysis: An empirically based method for automated software process discovery. Technical Report, 09-09, CSDL, 2009. http://csdl.ics.hawaii.edu/techreports/09-09/09-09.pdf

[31] Valle A, Portela E, Loures ER, Cestari JM. A framework for applying process mining techniques in software process assessments. In: Proc. of the IIE Annual Conf. 2014. 1339−1347.

[32] Kindler E, Rubin V, Schäfer W. Incremental workflow mining based on document versioning information. In: Li M, Boehm B, Osterweil LJ, eds. Proc. of the Unifying the Software Process Spectrum (SPW 2005). LNCS 3840, Berlin, Heidelberg: Springer-Verlag, 2006. 287−301. [doi: 10.1007/11608035_25]

[33] Jans M, Werf JMVD, Lybaert N, Vanhoof K. A business process mining application for internal transaction fraud mitigation. Expert Systems with Applications, 2011,38(10):13351−13359. [doi: 10.1016/j.eswa.2011.04.159]

[34] Castillo RP, Weber B, Pinggera J, Zugal S, Guzman GRD, Piattini M. Generating event logs from non-process-aware systems enabling business process mining. Enterprise Information Systems, 2011,5(3):301−335. [doi: 10.1080/17517575.2011.587545]

[35] Poggi N, Muthusamy V, Carrera D, Khalaf R. Business process mining from E-commerce Web logs. In: Daniel F, Wang J, Weber B, eds. Proc. of the 11th Int'l Conf. on Business Process Management. LNCS 8094, Berlin, Heidelberg: Springer-Verlag, 2013. 65−80. [doi: 10.1007/978-3-642-40176-3_7]

[36] Leemans M, van der Aalst WMP. Process mining in software systems: Discovering real-life business transactions and process models from distributed systems. In: Proc. of the 18th Int'l Conf. on Model Driven Engineering Languages and Systems (MODELS). ACM/IEEE, 2015. 44−53. [doi: 10.1109/MODELS.2015.7338234]

[37] Lohmann N, Verbeek E, Dijkman R. Petri net transformations for business processes—A survey. Trans. on Petri Nets and Other Models of Concurrency II, 2009,5460:46−63. [doi: 10.1007/978-3-642-00899-3]

[38] Cortadella J, Kishinevsky M, Lavagno L, Yakovlev A. Deriving Petri nets from finite transition systems. IEEE Trans. on Computers, 1998,47(8):859−882. [doi: 10.1109/12.707587]

[39] Lopezgrao JP, Merseguer J, Campos J. From UML activity diagrams to stochastic Petri nets: Application to software performance engineering. Workshop on Software and Performance, 2004,29(1):25−36. [doi: 10.1145/974044.974048]

[40] Reisig W. Petri Nets: An Introduction. Berlin, Heidelberg: Springer-Verlag, 1985. [doi: 10.1007/978-3-642-69968-9]

[41] Yuan CY. Petri Net Applications. Beijing: Science Press, 2013 (in Chinese).

[42] Zhu R, Li T, Mo Q, Dai F, Gao TL, He Y, Sun X. Heuristic parallelized mining single firing sequence. Computer Integrated Manufacturing Systems, 2016,22(2):330−342 (in Chinese with English abstract).

[43] Yang JW, Chen XO. A semi-structured document model for text mining. Journal of Computer Science and Technology, 2002,17(5): 603−610. [doi: 10.1007/BF02948828]

[44] Yang JW, Cheung WK, Chen XO. Learning element similarity matrix for semi-structured document analysis. Knowledge and Information Systems, 2009,19(1):53−78. [doi: 10.1007/s10115-008-0138-2]

[45] Yoon JP, Raghavan V, Chakilam V. BitCube: A three-dimensional bitmap indexing for XML documents. In: Proc. of the 13th Int'l Conf. on 2001 Scientific and Statistical Database Management. 2001. 158−167. [doi: 10.1023/A:1012861931139]

[46] Nayak R, Xu S. XCLS: A fast and effective clustering algorithm for heterogenous XML documents. In: Proc. of the Knowledge Discovery and Data Mining. 2006. 292−302. [doi: 10.1007/11731139_35]

[47] Tran T, Nayak R, Bruza P. Combining structure and content similarities for XML document clustering. In: Proc. of the 7th Australasian Data Mining Conf., Vol.87. ACM Press, 2008. 219−226.

[48] Algergawy AAA. Management of XML data by means of schema matching [Ph.D. Thesis]. Otto von Guericke University Magdeburg, 2010.

[49] Bezdek JC. Pattern Recognition with Fuzzy Objective Function Algorithms. New York: Springer US, 1981. [doi: 10.1007/978-1-4757-0450-1]

© 中国科学院软件研究所 http://www.jos.org.cn

[50] Pal NR, Pal SK. Entropy: A new definition and its applications. IEEE Trans. on Systems, Man and Cybernetics, 1991,21(5): 1260−1270. [doi: 10.1109/21.120079]

[51] Hung WL, Yang MS. Similarity measures of intuitionistic fuzzy sets based on Hausdorff distance. Pattern Recognition Letters, 2004,25(14):1603−1611. [doi: 10.1016/j.patrec.2004.06.006]

[52] Singh VP, Asce F. Hydrologic synthesis using entropy theory: Review. Journal of Hydrologic Engineering, 2011,16(5):421−433. [doi: 10.1061 /(ASCE)HE.1943-5584.0000332]

[53] De Medeiros AKA, Dongen BFV, Weijters AJMM. Process mining: Extending the α-algorithm to mine short loops. BETA Working Paper Series, WP 113, Eindhoven: Eindhoven University of Technology, 2004.

[54] Leemans SJJ, Fahland D, van der Aalst WMP. Discovering block-structured process models from incomplete event logs. In: Ciardo G, Kindler E, eds. Proc. of the Int'l Conf. on Applications and Theory of Petri Nets and Concurrency. LNCS, Cham: Springer-Verlag, 2014. 311−329. [doi: 10.1007/978-3-319-07734-5_6]

[55] Alves WMPVD, De Medeiros AKA, Weijters AJMM. Genetic process mining. In: Ciardo G, Darondeau P, eds. Proc. of the Applications and Theory of Petri Nets. LNCS 3536, Berlin: Springer-Verlag, 2005. 48−69. [doi: 10.1007/11494744_5]

[56] Zaki MJ. SPADE: An efficient algorithm for mining frequent sequences. Machine Learning, 2001,42(1-2):31−60. [doi: 10.1023/A:1007652502315]

[57] Zimmermann A. Understanding episode mining techniques: Benchmarking on diverse, realistic, artificial data. Intelligent Data Analysis, 2014,18(5):761−791. [doi: 10.3233/IDA-140668]

[58] Peterson JL. Petri Net Theory and the Modeling of Systems. Prentice-Hall, 1981.

附中文参考文献:

[12] 郁抒思,周水庚,关佶红.软件工程数据挖掘研究进展.计算机科学与探索,2012,6(1):1−31. [doi: 10.3778/j.issn.1673-9418.2012.01.001]

[16] 何克清,李兵,马于涛,黄贻望.大数据时代的软件工程关键技术.中国计算机学会通讯,2014,10(3):8−18.

[17] 王千祥,梅宏.大数据与软件工程.中国计算机学会通讯,2014,10(3):6−7.

[41] 袁崇义.Petri 网应用.北京:科学出版社,2013.

[42] 朱锐,李彤,莫启,代飞,高提雷,何云,孙雪.启发式并行化单触发序列挖掘算法.计算机集成制造系统,2016,22(2):330−342. [doi: 10.13196/j.cims.2016.02.006]

朱锐(1987－),男,山东临沂人,博士,讲师,CCF 专业会员,主要研究领域为软件过程,过程挖掘.

何臻力(1987－),男,博士,讲师,主要研究领域为云计算,大数据.

李彤(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件过程,形式化方法.

于倩(1975－),女,博士,讲师,CCF 专业会员,主要研究领域为软件工程.

莫启(1986－),男,博士,讲师,主要研究领域为业务过程管理.

王一荃(1983－),女,博士生,助理研究员,主要研究领域为软件过程,Petri 网.

© 中国科学院软件研究所 http://www.jos.org.cn