ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.3, March 2009, pp.524−545
doi: 10.3724/SP.J.1001.2009.03432

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

软件过程建模方法研究*

李明树[1,2+], 杨秋松[1,3], 翟 健[1,3]

[1](中国科学院 软件研究所 互联网软件技术实验室,北京 100190)

[2](中国科学院 软件研究所 计算机科学国家重点实验室,北京 100190)

[3](中国科学院 研究生院,北京 100049)

Systematic Review of Software Process Modeling and Analysis

LI Ming-Shu[1,2+], YANG Qiu-Song[1,3], ZHAI Jian[1,3]

[1](Laboratory for Internet Software Technologies, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[2](State Key Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[3](Graduate University, The Chinese Academy of Sciences, Beijing 100049, China)

+ Corresponding author: E-mail: mingshu@iscas.ac.cn

Li MS, Yang QS, Zhai J. Systematic review of software process modeling and analysis. *Journal of Software*, 2009,20(3):524−545. http://www.jos.org.cn/1000-9825/3432.htm

Abstract: Nowadays it has been widely accepted that the quality of software highly depends on the process that is carried out in an organization. As part of the effort to support software process engineering activities, the research on software process modeling and analysis is to provide an effective means to represent and analyze a process and, by doing so, to enhance the understanding of the modeled process. In addition, an enactable process model can provide a direct guidance for the actual development process. Thus, the enforcement of the process model can directly contribute to the improvement of the software quality. In this paper, a systematic review is carried out to survey the recent development in software process modeling. 72 papers from 20 conference proceedings and 7 journals are identified as the evidence. The review aims to promote a better understanding of the literature by answering the following three questions: 1) What kinds of paradigms are existing methods based on? 2) What kinds of purposes does the existing research have? 3) What kinds of new trends are reflected in the current research? After providing the systematic review, we present our software process modeling method based on a multi-dimensional and integration methodology that is intended to address several core issues facing the community.

Key words: software process; modeling; analysis; systematic review

摘 要: 通过软件开发实践,人们逐步地认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.软件过程建模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,同时,可执行的(enactable)软件过程模型可以直接指导实际软件开发活动,进而规范软件开发行为并最终提高软件质量.为了

* Supported by the National Natural Science Foundation of China under Grant Nos.60573082, 90718042 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant Nos.2006AA01Z185, 2007AA010303 (国家高技术研究发展计划(863)), the National Basic Research Program of China under Grant No.2007CB310802 (国家重点基础研究发展计划(973))

Received 2008-04-03; Accepted 2008-07-09

系统地了解软件过程建模方法研究的现状和最新进展,采用系统评价(systematic review)方法对该领域最近 10 年的主要研究进行了概括和分析.从一系列的相关研究中,选出来自 20 个会议和 7 种期刊的 72 篇文献,作为系统评价的依据.该系统评价回答了如下关于软件过程建模方法的 3 个问题,以便从总体上概括和把握该领域的研究:1) 软件过程建模方法主要基于什么范式;2) 软件过程建模方法研究的主要目的集中在哪些方面;3) 软件过程建模方法的研究有哪些新的趋势.同时,在仔细回顾和分析软件过程建模领域研究现状的基础上,给出了一种多维度的集成化软件过程建模方法.该方法有助于解决过程建模领域所面临的主要问题.

关键词: 软件过程;建模;分析;系统评价

中图法分类号: TP301 **文献标识码**: A

软件过程(software process)是指用于开发和维护软件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[1].通过软件开发实践,人们逐步地认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],即生产高质量的软件需要有一个高质量的软件过程.由于影响软件开发的各种因素,比如商业环境、开发技术以及开发人员,总是在持续不断地变化,因此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],而软件过程改进也构成了软件过程管理活动的核心.

为了支持软件过程的改进,研究者们提出了过程改进的不同周期模型,其中,Dion[5]提出了一个非常典型的三阶段过程改进模型,用于指导具体的过程改进活动.如图 1 所示,过程制定(process stabilization)是该周期模型的第一个阶段.在该阶段将描述所要执行的过程,并通过特定的方式发布过程的文档,同时确保所发布的过程在整个组织内得到执行;在随后的过程控制(process control)阶段,通过相应的工具支持项目数据的收集,在这些数据的基础上决定如何控制过程的执行;在最后的过程变更(process change)阶段,根据度量和分析的结果决定从哪些方面改进所执行的过程,在实施新的过程之前可能在小范围内进行实验和确认.此时,软件过程改进活动完成了一个周期,而后进入下一个改进周期.与此相类似,Humphrey 指出软件过程改进活动包含如下 3 个方面[4]:过程定义(process definition),需要明确地定义所要执行和改进的过程,包括清楚地定义过程内所有的活动、活动执行顺序以及活动的出口标准等;过程使用(process use),通过使用所定义的过程,在过程执行中发现一些改进机会;数据收集和分析(data collection and analysis),通过收集和分析过程执行时的数据,确定软件过程中的问题并决定如何进行改进.其他过程改进周期模型包括 PDCA(Plan/Do/Check/Act)和 QIP(quality improvement paradigm)以及描述了更多组织层次内容的模型 IDEAL 和 ISO 15504-7 等[6].

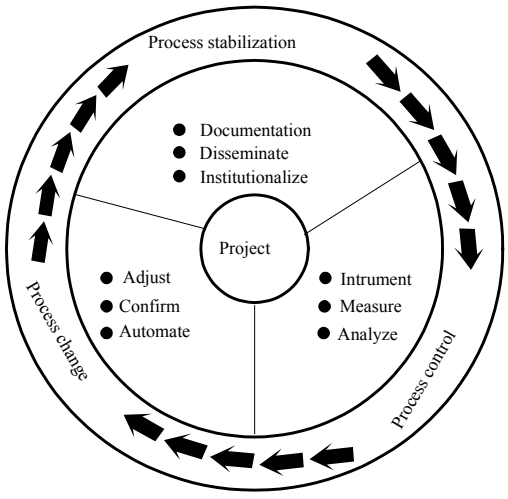

Fig.1 A typical software process improvement lifecycle model[5]

图 1 一个典型的软件过程改进生命周期模型[5]

从软件过程改进的周期模型可以看出,软件过程改进覆盖了从过程定义、执行、分析到过程变更(或者演

化,evolution)的一系列过程管理活动,因此,Kinnula[6]直接称软件过程改进为软件过程工程(software process engineering).对于软件过程改进的研究,主要分为两类[7].第 1 类是软件工业界所关注的、以能力成熟度集成模型(capability maturity model integration,简称 CMMI[3])为代表的软件过程评估(assessment)和改进模型,这些模型主要用于评估一个开发组织的能力级别或者成熟度级别,同时为开发组织提供一个可遵循的过程改进途径.第 2 类是研究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.与软件过程评估和改进模型不同,软件过程建模主要是为具体的过程改进活动提供方法和工具上的支持,覆盖了包括定义、执行、分析和变更在内的整个过程改进生命周期.

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,简称 PSEE)[7]展开的.一种建模方法所具备的描述、分析、执行和演化的能力主要依赖于所使用的建模语言,而 PSEE 决定了一种建模方法对实际开发活动所能提供的支持;PSEE 和过程建模语言往往是密不可分的,每个 PSEE 具有相关联的一种或者几种建模语言,而一种建模语言需要在相应的 PSEE 中被解释和执行.PSEE 的出现可以追溯到 20 世纪 70 年代,主要是通过数据流集成的方式,将一些原本孤立的开发工具组合在一起,比如需求分析工具的输出作为设计工具的输入、设计工具的输出作为代码生成工具的输入等等,而真正将软件过程作为一个实体进行支持的 PSEE,则是在 20 世纪 80 年代后开始出现[8].在 90 年代前后,特别是基于软件过程也是软件(software processes are software too)[9,10]的思想提出后,研究者们提出了多种 PSEE 和软件过程建模语言[7,8,11−13].

上述的建模方法一般是在软件建模和分析方法的基础上,针对软件过程建模的需要,作适当的改进而提出来的.但在研究过程中,人们逐渐认识到现有的方法在实际应用中存在着如下两个主要问题:a) 实际软件开发过程涉及很多要素,而软件过程模型作为实际软件开发过程的抽象,一般只描述了过程的某些方面.另外,由于开发人员所具有的主观性、软件开发活动所具有的创造性以及实际软件开发过程在执行中将会随着项目的进展或者外部组织环境的变化而不断变化,使得实际软件开发活动很少可以严格按照在 PSEE 中实例化的软件过程模型执行.因此,所构造的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].b) 在 PSEE 和建模语言的设计过程中,人们需要权衡相互矛盾的需求[15].比如,为使不具有太多工程背景的涉众(stakeholder)也能很好地理解软件过程,需要建模语言的表现形式较易于理解:一般需要图形化支持且不需要涉及太多的细节;而软件过程的执行和分析需要建模语言能够描述必要的细节:一般需要具有明确的操作语义,而且最好采用形式化的方法.如何能够满足这些相互矛盾的需求,是软件过程建模语言和 PSEE 设计中需要仔细权衡的问题,同时也是软件过程建模方法研究所一直面对的课题.

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.同时,对于分布式和全球协同环境下的软件过程建模和软件过程建模技术在企业的应用、实施也积累了一些经验.为了概括这一领域的最新进展,本文采用系统评价方法对该领域最近 10 年发表的论文进行了搜集、整理、概括和分析,期望可以有助于了解该领域所关注的主要研究问题以及所取得的进展,并对今后的研究具有借鉴和启发意义.

本文第 1 节介绍软件过程建模的基本内容.第 2 节介绍系统评价方法及其在软件过程和软件工程中的应用,并详细介绍本文所使用的系统评价过程.第 3 节给出软件过程建模方法的系统评价结果和相应的分析.第 4 节给出一种多维度的集成化软件过程建模方法.第 5 节是本文的结论.

1 软件过程建模

软件过程建模的主要目的是建立软件过程的抽象模型,通过对该抽象模型的分析增加对过程本身的理解和认识,从而可以更好地实施软件开发活动.对于同一个软件过程,所建立的抽象模型与建模方法、建模目的密

切相关.比如,对于支持控制流描述的建模语言,其相应的模型将会以过程中的一系列开发活动作为主线;而如果一个建模语言主要通过制品间的转换关系和出入口标准来描述一个软件过程,则相应的模型更主要的是描述开发活动中的制品.就建模目的而言,如果建模只是为了增加对过程的理解,所建立的模型只需比较高的抽象层次上对软件过程进行描述;而为了支持后续的软件过程执行或者更为详尽的分析,则需要过程模型包含必要的细节.一般认为,一个软件过程模型需要描述图 2 中所包含的元素及其相互间的关系[2].而建立抽象过程模型并不是软件过程建模的终点,在抽象模型的基础上,我们可以进行一系列的分析:软件过程模型一致性、正确性的检测,可以保证模型符合我们的预期;通过软件过程仿真,可以预测过程的性能或者分析不同的实施策略对整个过程的影响;通过比较两个过程模型之间的一致性,可以判断项目的实施过程是否与组织的标准过程相一致等.同时,对于软件过程模型更新策略和方法的研究,可以支持过程模型的灵活更新,以适应组织内部和外部环境的不断变化;对软件过程模型执行的研究,可以支持对实际软件开发活动的直接指导.软件过程建模研究的另外一项任务,是为上述的软件过程工程活动提供工具支持,特别是 PSEE.

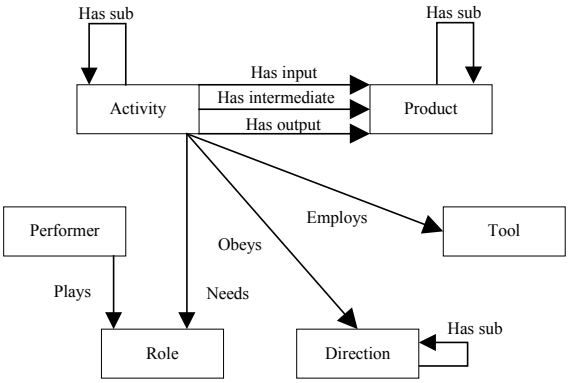

Fig.2 Basic information of a process model[11]

图 2 软件过程模型的基本内容[11]

软件过程建模影响实际开发活动的方式,即采用什么样的方式将这些过程技术引入到一个组织的实际管理、支持和工程活动中去,主要有两种.第 1 种方式是采用指南或者参考手册的形式.对于所要实施的软件过程,首先通过某种语言对其进行描述,制作成指南或者参考手册并分发给项目成员.项目成员在执行具体的开发任务时,将根据对指南的理解,以自认为正确、合理的方式完成指定的任务.这种方式的缺点是过分依赖于项目成员自己的理解、自觉和职业道德,所定义的过程模型对于实际开发活动并不存在很强的约束能力.因此,在实际应用过程中,众多软件开发组织投入了很多的人力和物力来定义过程文档,但其并没有发挥所期望的作用,而是被束之高阁.第 2 种方式是通过 PSEE 支持实际开发活动.在 PSEE 的支持下,软件过程工程活动可以划分为如下 3 个领域[14]:过程模型(process model)、虚拟过程执行(process enactment)和实际过程执行(process performance).这 3 个领域分别定义为:

(1) 过程模型(process model).该领域主要是由过程设计人员(process designer)采用特定的软件过程建模语言建立软件过程模型,同时对这些过程模型进行有效的管理.

(2) 虚拟过程执行(process enactment).该领域主要是由 PSEE 的过程引擎解释和执行预定义软件过程模型.

(3) 实际过程执行(process performance).该领域是指由人力或者非人力资源(如计算机)进行的实际开发活动.

如图 3 所示,3 个领域之间通过如下方式进行交互,并构成了软件过程工程活动的核心.

(1) 过程模型和虚拟过程执行(标记 1).实例化的软件过程模型被送入到 PSEE,由过程引擎对其进行解释和执行.其中,实例化是指确定模型中的一些参数,比如具体的时间、资源以及一些与项目相关的信息.

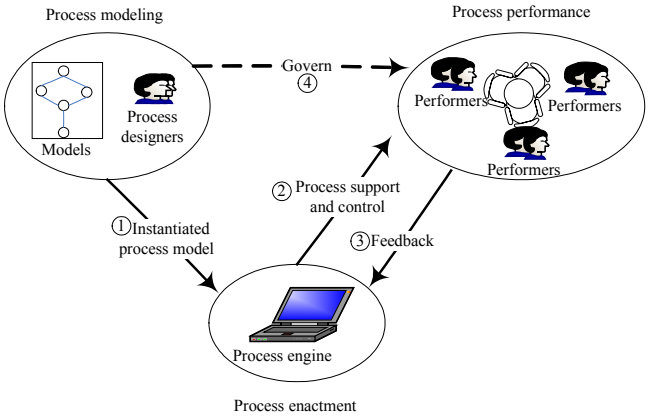

Fig.3 Three domains of software process engineering[7,14,16]

图 3 软件过程工程的 3 个领域[7,14,16]

(2) 虚拟过程执行和实际过程执行(标记 2 和标记 3).通过过程引擎对实例化过程模型的解释和执行,可以指导实际软件开发活动:向项目成员提供可以执行的活动列表、自动运行所需的开发环境,以及完成某些可以自动执行的任务.另一方面,当项目成员完成任务后需要向 PSEE 反馈相应的信息,以使得 PSEE 及时修改正在运行的过程模型的状态,并更新提供给项目成员的指导信息.

(3) 过程模型和实际过程执行(标记 4).通过 PSEE 的支持使实际软件开发活动受预定义软件过程模型的支配.

但这种方式在具体的实施过程中,也面临着如下问题:

(1) 信息反馈中的人为因素[14].由于虚拟过程执行和实际过程执行之间采用比较松散的耦合方式,由后者向前者的信息反馈需要人的参与,不可避免地由于人为因素的存在,使得反馈的信息不准确、不及时,可能为了取得某种结果,而引入主观的甚至是错误的信息.

(2) 任务抽象层次不同[16].软件过程模型任务定义的抽象层次,相对于实际的开发任务而言,仍显得太高.模型中的任务可能是修改需求、体系结构设计、模块编码、书写测试用例等,但实际开发中需要一系列的工作来完成过程模型中的一项任务.由于抽象层次的不同,软件过程模型往往只是描述了实际开发活动的一个很小的子集,使得很多实际开发活动不受 PSEE 的控制,从而无法真正规范整个实际软件开发过程.

(3) 产品描述的粒度不同[16].在软件过程模型中,一般以整个文档作为活动的输入、输出参数,而在实际的开发活动中,更多涉及到的是文档的某一部分,比如需求项、设计文档的某个模块、某个测试用例等.

由于上述问题的存在,导致虚拟执行的软件过程和实际执行的软件过程之间经常产生严重的偏差,以至于 PSEE 对于实际的开发活动失去了支持和指导的作用.

2 系统评价方法

传统的综述(narrative or ad hoc review)一般是对部分研究结果的定性概括,同时在研究过程中可能为了验证预期的结果而或多或少地引入一定的主观性,同时并没有将整个研究过程记录下来以供其他研究人员对该项研究本身进行评价.而在实施系统评价时,需要根据所研究的问题制定系统评价的实施规范,该规范包含了特定的策略以实现全面和彻底的文献搜索;明确的准则以决定如何取舍搜索到的文献;评估每项研究的质量准则;对获得的数据进行综合和分析的方法(比如元分析,meta-analysis).同时,对于需要人为参与的步骤(文献的取舍、每项研究结果的评价等),需要综合多名系统评价人员的结果,以使得评价尽可能的客观和公正.

具体来讲,一个系统评价的实施一般包含如下几个基本步骤[17,18]:

(1) 明确所关注的研究问题.即通过该系统评价最终所要回答的问题,同时可能需要进一步明确所感兴趣的研究方法(intervention)和研究结果(outcome)等其他限定条件.

(2) 确定相关研究的搜索策略.确定需要查询的搜索引擎、会议论文集和期刊,根据所要研究的问题和已有文献的关键词确定查询语句,而后通过该语句查询所确定的引擎、论文集和期刊.根据查询结果,确定是否需要进行二级查询(比如引入新的关键词或者添加新的论文集和期刊).

(3) 确定哪些研究被系统评价所采纳.所有参与系统评价的人员根据共同的标准对于上一步搜索到的每项研究(论文)进行质量评价,以决定其是否被包含到系统评价中,作为支持最终结论的证据.若不同的评价人员对同一研究给出了不一致甚至相互矛盾的结果,则需要对这些结果进行综合考虑或者采取最终协商一致的方式予以解决.

(4) 数据提取和分析.对于被采纳的每项研究,根据系统评价的需要提取相应的数据,而主观的数据项则需要多个系统评价人员的参与,而后需要对所获得的数据进行综合和分析,并对结果给出解释.

(5) 完成报告.在上述工作的基础上完成最终的报告.

系统评价方法在循证医学(evidence-based medicine)中使用得非常广泛[18],最近几年,该方法也被逐渐地应用到了软件工程领域.根据 3 个用于指导在医学领域内如何实施系统评价的指导手册,Kitchenham[17]定义了一个在软件工程领域内实施系统评价的规程(procedure),该规程所包含的步骤与一般系统评价所包含的步骤基本相同,只是用软件工程中的一些场景代替了原来医学领域的内容.同时,由于与医学领域相比,软件工程在经验研究方面相对比较缺乏且不是很严格(缺乏量化的结果),而在实施过程中省略了用于合并不同量化研究结果的统计分析方法.Hannay 等人[19]从 5 453 篇于 1993 年~2002 年发表于期刊和会议上的文章中,选出了 103 篇用于从总体上分析理论的使用情况.通过该系统评价,作者们发现其中的 24 篇文献使用了 40 种理论,其中约有三分之一是由文章的作者自己提出来的,所使用的理论具有很强的跨学科特性;但在经验软件工程中理论驱动的研究还比较少.Kitchenham 等人[20]使用系统评价,试图比较组织内和组织间的成本模型估算的准确性.从所搜索到的论文中,作者们最后选出了 10 项相关的研究,有 7 篇论文给出了明确的结论,其中 3 篇支持这两种估算模型没有显著的区别,而另外 4 篇支持组织间的模型要明显的比组织内的模型要差.作者们最后发现,根据现有的研究很难得到一个明确的结论,同时也发现组织内估算模型优于组织间估算模型的结论,一般是在较小的数据集上得到的.Jörgensen 等人[21]从 74 本期刊中选出了 304 篇文献,试图回答关于软件开发成本估算领域研究现状的 8 个问题:比如哪些期刊发表了软件成本估算的文章、哪种期刊在软件成本估算领域最具影响力及其论文主题所具有的倾向性、哪种估算方法最为人们所关注,以及随着时间的发展有什么样的变化等.

对于本文关于软件过程建模方法的系统评价,相应的每个步骤如下:

(1) 研究问题.通过该系统评价,我们尝试着回答如下问题:

问题 1:软件过程建模方法主要基于什么范式?

问题 2:软件过程建模方法研究的主要目的集中在哪些方面?

问题 3:软件过程建模方法的研究有哪些新的趋势?

(2) 文献搜索.我们首先采用如下关键词“software process” AND(“model” OR “language” OR “representation”),在 Google Scholar 中搜索发表时间为 1998 年(包括 1998 年)以后的文章,共有 15 700 项相关记录(2008 年 2 月 3 日),而 Google Scholar 只提供最相关的前 1 000 项.对于其中的每一项,我们根据题目和摘要查看其是否与该系统评价的主题相关,若相关则获得该论文的 BibTex 引用并录入 JabRef(支持 BibTex 的引用管理软件,http://jabref.sourceforge.net/).类似地,我们将上面的查询条件转换为 IEEE Xplore 可以接受的输入形式并进行查询,共搜索到 550 项相关记录(2008 年 2 月 4 日),类似地,IEEE Xplore 只提供最相关的前 500 项,对搜索到的内容也进行上述的处理.由于无论采用何种搜索方法和策略,查全率(recall)的提高意味着查准率(precision)的下降,反之亦然.为了使评价的证据更充分和准确,根据上面搜索的结果和我们的经验,采用人工的方式,逐年检查了于 1998 年后发表在主流会议和期刊中的论文,以避免漏掉高水平的相关研究.

(3) 入选标准.对于在上一步通过阅读题目和摘要过滤后的论文,在阅读其内容的基础上依据如下的准则决定是否将其作为该系统评价的证据:对于短文或者 Poster,通过 DBLP 查询作者是否有相关的工作发表在其他地方,若有,则引用相对比较完整的论文,若没有,则将相应的短文或者 Poster 排除掉;若论文的目的是提出一种

建模语言,则将该论文包含进来;虽然论文的目的不是提出一种建模语言,但是基于已有的一种建模语言提出了新的软件过程建模技术和方法(比如支持过程的表示、执行、分析、验证、仿真、演化、一致性检测等),则也将该论文包含进来;虽然论文中没有涉及到具体的语言,但提出了关于软件过程建模新的方法论,同样也将这类论文包含进来;对于与商业过程(business process)和工作流(workflow)相关的文章,我们只选择了那些在背景中明确提到软件或者软件开发的文章.

同时对于内容重复的文章:同一作者类似的文章发表在多个地方或者一篇综述很好地概括了一系列的相关研究,只选择了那些我们认为更具代表性的文章.表 1 和表 2 分别列出了最终入选的会议和期刊论文,其中会议论文 45 篇,期刊论文 27 篇.根据论文引用次数,图 4 和图 5 分别给出了引用 10 次以上(包含 10 次)和发表后每年被引用 2 次以上(包括两次)的论文;图 6 和图 7 分别给出了入选的会议和期刊论文的年份分布情况.

(4) 数据提取与分析.对于软件过程而言,由于软件开发本身与很多的因素相关,实验中很难使自变量和因变量以外的其他因素做到可控,还存在软件开发的不可重复性问题,因此,一般情况下很难定量地评估软件开发中某一个局部或者方面的改善对整体的影响.在软件过程建模领域,虽然已经提出了很多种软件过程建模方法,由于每种建模方法提出的目的、所具有的理论基础、所能解决的问题和应用的场合不尽相同,因此据我们所知,没有定量的比较建模方法的研究.所以,该系统评价的数据提取和分析变得相对比较简单,我们只是对入选的论文根据提出的问题进行分类、分析和总结.

(5) 完成报告.与系统评价方法中所推荐的模式相类似,我们首先在该部分概要介绍了系统评价方法,并给出了本文所使用的系统评价过程,下一节我们将给出该系统评价的分析结果.

Table 1 List of selected conference papers

表 1 入选会议论文列表

Conferences	Number	Papers
ACM Symposium on Applied Computing	1	[22]
ACS/IEEE International Conference on Computer Systems and Applications	1	[23]
Annual Hawaii International Conference on System Sciences	1	[24]
Annual IEEE/NASA on Software Engineering Workshop	1	[25]
Annual International Computer Software and Applications Conference	1	[26]
Asia-Pacific Software Engineering Conference	10	[27−36]
Australian Software Engineering Conference	1	[37]
EUROMICRO Conference	2	[38,39]
European Workshop on Software Process Technology	7	[15,40−45]
International Workshop on Software Processes, International Software Process Workshop and International Workshop on Software Process Simulation and Modeling, International Conference on Software Process	8	[46−53]
IEEE International Conference on Automated software engineering	1	[54]
IEEE International Enterprise Distributed Object Computing Conference	1	[55]
IEEE Symposia on Human Centric Computing Languages and Environments	1	[56]
International Conference of the Chilean Computer Science Society	1	[57]
International Conference on Software Engineering	2	[58,59]
International Conference on Software Engineering Advances	2	[60,61]
International Symposium on Empirical Software Engineering	1	[62]
International Workshop on Database and Expert Systems Applications	1	[63]
The Unified Modeling Language. Beyond the Standard	1	[64]
Workshop on Enabling Technologies	1	[65]

Table 2 List of selected journal papers

表 2 入选期刊论文列表

Journals	Number	Papers
ACM Transactions on Software Engineering and Methodology	4	[16,66−68]
Automated Software Engineering	2	[69,70]
Information and System Technology	11	[71−81]
Journal of System and Software	4	[82−85]
ACM SIGSOFT Software Engineering Notes	3	[86−88]
IEEE Transactions on Software Engineering	1	[89]
Software Process: Improvement and Practice	2	[90,91]

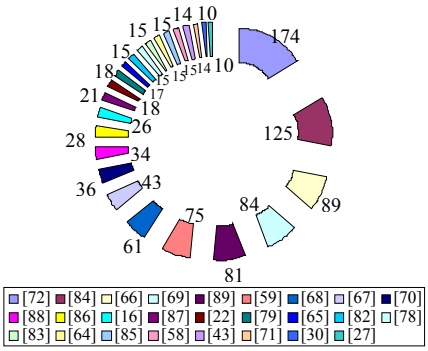

Fig.4 Papers referred not less than 10 times
图 4 引用次数不少于 10 次的论文

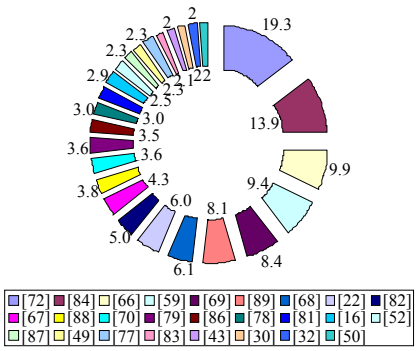

Fig.5 Papers referred not less than 2 times every year
图 5 发表后每年引用次数不少于 2 次的论文

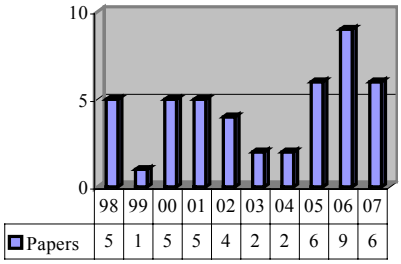

Fig.6 Statistics for publishing years of conference papers
图 6 会议论文发表年份统计

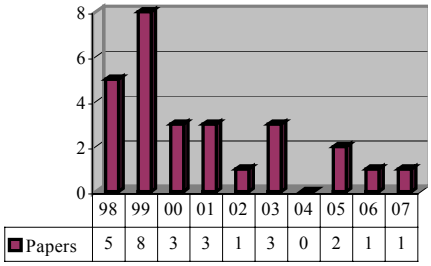

Fig.7 Statistics for publishing years of journal papers
图 7 期刊论文发表年份统计

3 评价结果和分析

3.1 问题1:软件过程建模方法主要基于什么范式

软件过程技术的最终目的是实现通过软件过程模型来驱动实际软件开发活动,以使得软件开发活动变得可控和可预测[2],而实现该目标的前提是能够通过一定的表示形式,准确地描述和把握所要实施的软件过程.从不同的角度和背景出发,研究者们提出了多种软件过程建模方法.软件过程建模方法的范式(paradigm)是指软件过程建模方法的基本类型,即通过什么方式对现有的软件过程建模方法进行分类.通过回答软件过程建模方法主要基于什么范式这一问题,我们可以了解该领域人员研究软件过程的基本方式;通过对比和分析,可以发现哪一种范式为更多的人所接受,或者能够解决更多为人们所关注的问题.

为了对软件过程建模方法进行系统的分析,人们提出了多种分类方式.根据建模方法的基本特点,Arboui 等人[7]将其按照如下 5 个类别进行划分:逻辑(logic)、过程(procedure)程序、人工智能(artificial intelligence)、触发器(trigger)、Petri 网和面向对象(object oriented).Montangero[2]根据建模方法与计算机科学其他领域的关系,则将建模方法划分为如下 7 个基本类型:项目管理(project management)、形式化规约语言(formal specification language)、非形式化设计表示方法(informal design notation)、编程语言(programming language)、数据库语言(database language)、工具集成机制(tool integration mechanism)、工作流和群件(workflow and groupware). Zmali[12,13]根据过程建模语言的技术特点和对表示、演化、虚拟执行、评估和人力资源的支持程度,对现有的建模方法进行了分类.

上述分类方法主要是根据软件过程建模中所使用的理论工具,对建模方法进行分类.其主要优点是通过类别信息可以初步地把握一种建模方法的基本机制;而缺点主要在于,建模方法所使用的理论不胜枚举,因此上述分类方法很难准确和客观地覆盖所有的建模方法;而在进行建模方法选择时,人们主要是从自身的建模需求出发,因此,上述分类方法也很难为方法选择提供有意义的指导.本文提出了一种新的分类方法,我们首先根据软

件过程的描述方式,将软件过程建模方法分为 4 个类别,每一类又划分为多种典型的实现方法,而后根据实现方法可以对表 1 和表 2 中具有明确建模方法的论文进行分类.

首先,将软件过程建模方法分为如下 4 个类别.同时,每个类别又包含了一些典型的实现方法:

(1) 静态关联的元素(statically connected elements).在基于该范式的建模方法中,软件过程被看作具有一定关系的元素(比如角色、活动和资源)的组合,而元素间的组成关系是静态的.这类建模方法适合于表达软件过程的静态方面和软件过程元素在概念层次的联系,一般不涉及软件过程的执行.比较典型的实现方法包括:面向对象的类图(class diagram,简称 CD)、UML 中的多种静态视图[92]、软件过程工程元模型(software process engineering metamodel,简称 SPEM)[93]、元对象设施(meta object facility,简称 MOF)[94]、本体(ontology)[95]、实体关系图(entity relation diagram,简称 ERD)[96]等.

(2) 顺序执行的活动(sequentially executed activities).在基于该范式的建模方法中,能够描述软件过程的执行,但所描述的是一组顺序执行行为.比较典型的实现方法包括 UML 状态图(state diagram,简称 SD)[92]、顺序关系(precedence relationship,简称 PR)[43]以及基于一般自动机的状态转换图(state transition diagram,简称 STD).

(3) 并发交互的活动(concurrently communicated activities).在基于该范式的建模方法中,软件过程由一系列的并发元素组成,每个元素都有自己的内部行为;而元素之间通过通信或者同步实现相互间的协调.比较典型的例子包括 UML 活动图(activity diagram,简称 AD)[92]、角色活动图(role activity diagram,简称 RAD)[97]、动态活动网(dynamic task net,简称 DTN)[98]、规约描述语言(specification and description language,简称 SDL)[99],以及可以描述并发系统的形式化方法,比如 Petri 网(Petri nets,简称 PN)、对象 Petri 网(object Petri net or nets-within-nets,简称 OPN)[100]、顺序通信进程(communicating sequential processes,简称 CSP)[101]、π演算(π-calculus)[102]等.另外,基于组件的软件过程建模(component based software process modeling,简称 CBSPM)、并发状态转换图(concurrent state transition diagram,简称 CSTD)也可以归为这一类别.

(4) 隐含的并发活动(implicitly concurrent activities).在基于该范式的建模方法中,软件过程的行为并没有被明确地表示出来,而是通过其他方式推导出来.软件过程的行为是具有一定智能的实体之间通过协商或者通过集中逻辑推理的方式来产生的,比较典型的实现方法包括多 Agent 系统(multi-agent system,简称 MAS)[103]和基于规则(rule)的过程执行[104].在软件过程仿真(software process simulation)[84]中,软件过程一般被看作是一个宏观系统,主要是通过过程的输入和输出来观察整个软件过程.比较典型的实现方法包括离散事件模拟(discrete event simulation,简称 DES)[105]和系统动力学(system dynamics,简称 SYDY)[106].

其次,根据每个类别中的具体实现方法,表 3 对表 1 和表 2 的论文进行了归纳.但对于不包括具体建模方法的经验报告、方法论和综述性质的文章,则没有包含在表 3 中.由于一种建模方法在其不同层次或者不同的分析阶段可能使用不同的范式,因此一篇论文中可能出现在多种范式中.

Table 3 Paradigms of software process modeling methods

表 3 软件过程建模方法范式

Paradigms	Papers
Statically connected elements	CD[28,38,43,57,64,68,69,76], UML[78,88], SPEM[33,35,53,82], MOF[55,28,43,61,63,64,77], Ontology[30], ERD[23], Others (Artifact Class[89], Private Graphic Notations[56,59], Alloy[80], AAG[27])
Sequentially executed activities	PR[38,43], STD[57,69,89], Others (Alloy[80])
Concurrently communicated activities	AD[28], RAD[78], DTN[42,88], SDL[44], CSPL[75], PN[33,67,70−72], OPN[36,48,60], CSP[53], π-calculus[47,51,79], CBSPM[54], CSTD[86,89], SPP[34,62], PP[31,73], Others (PARADIAGM[57], Communicating FSM[59], UML2.0 Action Package[55], Data Flow[69], Control Flow[69], Alloy[80])
Implicitly concurrent activities	MAS[22,39,52], Rule[23,56], DES[29,35,84], SYDY[35,84,85], Others(Hybrid Models[83,90], Markov Decision Theory[26]

下面,我们将分别论述每种范式中的主要实现方法以及每种实现方法的典型研究案例.

(1) 静态关联的元素.对于软件过程元素间静态关系的描述,人们主要采用非形式化的面向对象表示方法.Acuna 等人[57]通过面向对象类图中的“is-a”和“be a part”关联,表示项目中角色、活动、产品之间的静态关系.UML 中的多种静态视图也可以用于描述过程元素间的静态关系:Jäger 等人[88]使用 UML 包图(package

diagram)表示软件过程的结构;Odeh 等人[78]把用例(use case)作为过程模型和系统模型之间的桥梁,在两种模型之间建立联系.而对于描述模型语义和约束的元模型,也被广泛地用于描述软件过程元素之间的静态关系:Li[53]采用扩展的软件过程工程元模型,描述软件过程中活动、产品、角色之间的关系及其集成;Bendraou 等人[55]采用元对象设施描述建模语言 UML4SPM 的静态语义.此外,Ceravolo 等人[30]通过本体对 XP 中的概念和所涉及的实体进行建模,以支持对相关概念的理解和实体的检索;Ahmed-Nacer 等人[23]采用实体关系图支持过程元素及相互间关系的表示、存储.

(2) 顺序执行的活动.Cugola[89]采用基于有限状态自动机的状态转换图,表示过程中制品的状态以及状态间的转换;Acuna 等人[57]采用状态转换图,描述过程元素的内部和过程元素之间的顺序行为.Franch[38]和 Ribò[43]采用前趋关系表示活动间的前趋和后继关系.

(3) 并发交互的活动.支持该类过程描述的建模方法,一般与描述并发软件系统的形式化方法具有较为密切的关系,前者往往是在后者的基础上,根据软件过程建模的需求作适当的改进而提出来的.Podnar 等人[44]使用电信领域内广泛采用的规约描述语言,对软件过程进行描述、仿真和验证.Bendraou 等人[33]和 Ge 等人[60],分别采用 Petri 网、对象 Petri 网[100]描述 SPEM 2.0 的语义和支持软件过程的多视图建模.对于进程代数方法,Li[53]和 Yang 等人[47]分别采用顺序通信进程、π演算描述软件过程并发活动间的通信和同步.Cobleigh[86]采用并发的状态转换图描述 Little-JIL 的语义.

(4) 隐含的并发活动.通过隐含方式描述软件过程的研究,主要分为两大类.第 1 类是采用人工智能逻辑推理的方式,得到软件开发过程:Zhao 等人[22]采用多 Agent 系统,由具有一定感知、知识表示、推理和协商能力的 Agents,通过协作的方式建立软件开发过程,并完成开发任务;Ahmed-Nacer[23]把软件过程用一系列的逻辑规则来表示,只有当一条规则的条件部分被满足时,才会执行规则结论部分所对应的活动.同时,一个活动的执行可能会改变全局的状态,从而改变规则条件部分是否被满足的状态.第 2 类是软件过程仿真.在软件过程仿真中,一般也不具备一个明确的软件过程活动序列,而只是通过一些参数定义软件过程的仿真模型,仿真模型的每次运行则产生一个软件过程的实例.Padberg[26]使用离散事件系统进行软件过程仿真,以比较项目的不同调度策略.Pfahl[85]把系统动力学仿真与传统软件过程建模结合在一起,以降低系统动力学模型建立的难度.

从表 3 和上面的分析中,我们可以进一步得到如下结论:

表 3 中的前两种范式主要描述软件过程的静态方面或者顺序活动行为.由于这两种描述方式主要是基于一些非形式化的面向对象表示方法和描述顺序行为的图示,不涉及软件过程活动间的并发和交互,因此这类方法易于理解和使用,比较适合于以沟通、理解、提取、表示为目的的软件过程建模.另一方面,软件的开发特别是大规模软件或者分布式开发,是一项很复杂的过程.软件过程的很多任务都是并行的,同时,因为资源、信息或者数据的共享,需要活动之间相互协调和配合.因此,表 3 中后两种范式所具有的、描述软件过程行为并发和交互的能力,对于真实、全面地刻画一个软件过程来讲是至关重要的.

表 3 中的前 3 种范式都属于说明性(prescriptive or descriptive)软件过程模型的范畴,而在最后一种范式中,一般不具有一个明确的软件过程,或者建立软件过程模型的目的并不是为了指导实际软件开发活动.一个说明性的过程模型主要用于说明软件开发活动如何实施,包括主要的开发活动、活动的组织、活动的执行者、工作产品在活动间的传递、工作产品间的关系等等,同时可以从静态和动态、不同的抽象层次来描述一个预定义的或者实际软件开发过程.因此,从表 3 可以看出,软件过程建模方法的研究主要集中在说明性的软件过程模型.说明性过程模型的优点在于可以直接指导实际软件开发活动,并且与实际软件开发过程之间基本能够对应.对于基于多 Agent 的软件过程建模,其优点一般源于 Agent 所具有的自治性、与其他 Agent 和环境交互时所具有的灵活性,以及能够自主地推理、规划和执行等特点;但建立起支持多 Agent 所需的知识库是一项很具挑战性的任务,同时,实际软件开发活动中个体的组织方式与多 Agent 系统相比更具刚性.类似地,在基于规则的软件过程模型中,只有当一条规则的前置条件被满足时,才执行后置条件所对应的动作,其缺点在于缺乏一个总体上清晰和明了的软件过程.而软件过程仿真主要用于软件过程的随机模拟和分析,而所建立的模型只是在不影响分析结果的前提下对要研究的过程的粗略抽象,其目的并不是为了指导软件过程的执行或者刻画实际软件过程.

从表 3 可以看出,具体描述软件过程的方法有很多,但由于每种方法都具有各自的特点,及其所适合的描述软件过程的角度、方式和场合,因此很难比较具体的建模方法的优劣;同时,在实际应用中也没有哪种方法能够覆盖所有的软件过程建模需求,并表明比其他方法取得了更为广泛的应用.该问题导致一个软件开发组织在选择 PSEE 或者建模语言时将会变得很茫然,因为存在着太多的方法,且没有被大家公认的主流方法.

3.2 问题2:软件过程建模研究的主要目的集中在哪些方面

软件过程对于提高软件质量,增强一个软件开发组织的竞争力,具有重要的作用;而软件过程建模和分析有助于人们更好地理解正在实施或者将要实施的软件过程,规范实际软件开发活动,分析软件开发过程中潜在的问题,促进组织过程的不断改进.对于软件过程建模分析的目的,Gruhn[8],Zamli 等人[13]和 Arbaoui 等人[7]从各自的角度给出了各有侧重的论述,从总体上来讲可以把它们概括为如下 4 类:

(1) 文档化(documentation).软件过程的文档化主要是指采用适当的表达形式对软件过程进行描述,主要强调的是可理解性,一般不需要太多的细节或者只需要显示过程的主要内容即可.

(2) 执行(execution).软件过程的执行主要是指具有操作语义的软件过程模型,在 PSEE 的支持下被实例化、解释和执行.如图 3 所示,软件过程执行主要涉及到过程模型、虚拟执行和实际执行 3 个领域的交互.通过软件过程的执行,可以使得实际开发活动遵循预定义的过程模型.

(3) 分析(analysis).与其他领域的建模相同,软件过程建模的另一个重要目的是对软件过程进行分析,从而增加对过程本身的理解.

(4) 演化(evolution).由于软件过程所涉及的因素很多,软件过程所处的技术、组织和商业环境也在不断地改变,因此客观上需要对软件过程不断地进行更新,以反映环境的变化.演化主要关注软件过程模型更新的策略和方法,以使得软件过程模型能够被及时、灵活地更新,且对正在运行的过程的影响做到最小.

表 4 对于表 1 和表 2 中的论文,根据其软件过程建模的目的进行了分类.通过该分类,可以了解研究者们所试图解决的具体问题.在该表中,若一项研究出现在支持文档化这一行,是指过程文档化作为该研究的一项主要目的被明确地提了出来,而对于间接支持过程文档化的研究则没有划归这一类:比如提出了一种建模方法,该方法很显然地可以支持过程的文档化,但研究的主要目的是用该方法支持过程的执行.下面我们将详细分析每个类别的主要研究问题.

Table 4 Purposes of software process modeling

表 4 软件过程建模目的

Purposes	Papers
Documentation	[28,30,34,38,39,44,46,47,49,53−55,57,59−64,68−70,73,76,77,80]
Execution	[16,22,33,38,42,44,47−49,51,53,55−57,59,67,69−73,79,80,87−89]
Analysis	[27−29,32,35,36,47,49,50,66,70−72,74,80,82−86,90]
Evolution	[23,27,28,31,73,75,88]

(1) 文档化.表 4 中第 1 行列出的研究都将支持过程文档化作为一个目标,除了从一般意义上支持文档化以外,一些研究还从其他方面深化对文档化的支持:Bhuta 等人[46]和 Gary 等人[54]从基于组件的开发这一思想出发,讨论了如何通过过程组件支持过程片断的重用;Jaccheri 等人[68]提出了一种基于面向对象的建模语言,以支持软件过程的提取(elicitation);Fuggetta 等人[76]基于 E3 讨论了如何对比较复杂的、大规模的软件开发过程进行建模;Lee 等人[28]探讨了如何通过 UML 支持软件过程的各种视图;Ruiz 等人[63]通过 MOF 和 XMI 支持软件过程的交换和传递;Sellers 等人[77]提出了 SPEM,OPF,OOSPICE 和 LvieNet 这 4 个过程元模型的超集,以更好地支持软件过程建模;Baldassarre 等人[62]从过程模式的角度,讨论了如何支持过程模型的定制,以适应过程所执行的具体环境.

(2) 执行.如前所述,由于信息反馈缺乏客观性,任务抽象层次和产品描述粒度的不一致,导致实际软件开发活动并不能严格地与在 PSEE 中执行的虚拟过程保持一致,而实际情况一般是两者之间的偏离(deviation)经常很明显,以致于 PSEE 对于实际软件开发活动失去了指导意义.表 4 中第 2 行列出的每一项研究都将支持软件过程的执行作为其研究目的,而下面的论文还着重阐述了如何解决上面的偏离问题:Cugola[89]基于一个面向工作

制品(artifact)的语言 PROSYT,使得相应的 PSEE 可以在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在什么时候采用手工方式将其消除;Pohl 等人[16]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;Yang 等人[47]通过过程挖掘和模型检测的方法及时地检测软件执行过程中的偏离.

(3) 分析.对于表 4 中第 3 行所列出的研究可以进一步分为 3 类: a) 关注软件过程模型本身的正确性以确保该模型准确地描述了所希望表达的内容.Lee[28]提供了一种方法可以检查用 UML 表示的过程中的不一致(inconsistency),并可以在必要的时候消除这种不一致,而模型检测方法也分别被 Cobleigh 等人[86](有限状态自动机表达的安全性的验证)、Min 等人[71](基于 Petri 网的结构分析)、Yang 等人[47](μ演算表达的活动时序关系的检查)以及 Wallace[80](通过 SAT 求解找到不满足特定性质的反例)等用于检查软件过程模型是否满足给定的性质. b) 比较两个过程模型是否一致或者等价,而比较的对象与所要解决的问题密切相关.Yang 等人[47]、Cook 等人[66]和 Huo 等人[32]首先通过不同的过程挖掘方式得到实际开发过程的模型,而后比较该模型与预定义软件过程模型的一致性,从而得知预定义的软件过程是否在实际开发过程中被很好地贯彻,或者如何修改软件过程的定义以更好地描述实际开发情况.Yoon 等人[27]通过比较标准过程和根据项目需要裁减后的过程相似程度,确定裁减的合理性,而 Breu 等人[74]给出了一个比较两个模型一致性程度的分层模型.c) 通过软件过程仿真可以掌握一些关于软件过程的基本规律,Kellner 等人[84]对软件过程仿真进行了很好的概括.

(4) 演化.对于演化的支持可以分为在线(on-the-fly)和离线(offline)两种方式,前者支持在更新软件过程模型的同时,也更新已经被实例化且正在执行的过程,在后一种方式中只更新软件过程模型,而正在执行的过程仍按旧版本运行.为了支持软件过程的演化,一般需要过程建模语言具有反射(reflection)功能,即可以分析用自身代码写的过程片断,并动态地创建相应的实例,从而可以实现软件过程在执行时被动态地修改和替换.表 4 第 4 行所列的研究都支持软件过程的离线演化,而对于在线演化方式中如何使得正在执行的过程迅速地按修改后的过程运行,仍需进一步的研究,主要问题在于转换的过程中可能存在信息冗余和缺失的问题.

从表 4 和上面的分析中,我们可以进一步得到如下结论.

表 4 中,对于软件过程演化的研究相对而言较少,该现象与软件过程建模技术特别是 PSEE 的实际应用情况密切相关.现有关于软件过程演化的研究,更主要的是从建模方法本身出发,进行理论和实现方式上的探讨;但由于在实际开发环境中,软件过程建模技术与实际开发活动之间更多地是采用松散的组合方式,即尚未实现完全由软件过程模型驱动的开发,因此对于过程演化的需求并不是很迫切.

表 4 中,对于软件过程文档化、执行和分析的研究,虽然从文章数目上很难进行区分,但从我们的分析来看,软件过程分析在软件过程建模的研究中日益受到重视.主要原因在于:在过去几十年中,人们提出了大量的软件过程建模和表示方法,但也逐渐认识到,距离实现理想中完全的软件过程模型驱动开发,还有很长的路要走.而软件开发本身所具有的复杂性、不可重复性以及创造性,使得精确、完全地描述一个实际软件过程比较困难.在这种情况下,通过特定的方法对软件过程的某些方面进行描述,利用建模方法提供的手段对过程的特定方面进行分析或者进行过程仿真,增加对所建模过程的理解,可能更具有现实意义.

3.3 问题3:软件过程建模方法的研究有哪些新的趋势

在该部分,我们试图在软件过程的文档化、执行、分析和演化等这些传统研究主题之外,在对最近 10 年软件过程建模方法研究进行系统整理的基础上,找到一些新的研究热点.这些热点从某种程度上反映了软件过程建模研究的发展趋势,或者研究者们试图重新认识某些对该领域的发展来讲非常重要的问题.经过归纳和分析,主要是如下两个方面的研究问题:

分布式或者全球开发(global software development,简称 GSD).“随着持续、不可逆转的商业全球化,特别是软件密集型高科技业务的全球化和受经济因素的驱动,全球市场日益成为一个不可分割的整体,且产生了跨越传统国家边界的竞争和协作方式.与此相适应,从软件需求的提出到设计、开发、测试、发布以及维护等,整个过程也不可避免地打上全球分布式、协作开发的烙印”[107].通过分布式开发,软件开发组织可以充分地利用各地的人力资源,以更低的成本获得更为优秀的资源,更加接近客户和目标市场,及时把握客户需求的变化并作出响

应以及利用时差实现 24 小时开发以缩短开发周期等[107,108].

软件过程技术实施.软件过程技术的最终目的是通过预定义的软件过程模型驱动实际软件开发活动[2],因此,如何将软件过程建模技术应用到实际开发环境中是一个很重要的研究课题.作为软件过程建模方法研究的一部分,许多研究者提供了他们实际应用某些过程技术时所获得的经验,而这些经验有助于我们了解实际软件开发需要什么样的软件过程建模语言和 PSEE,以及如何使得一个开发组织能够更好地接受一项软件过程新技术.

表 5 列出了表 1 和表 2 中与上述两个趋势相关的论文.下面将分别论述与每个趋势相关的主要研究.

Table 5 Trends of software process modeling

表 5 软件过程建模趋势

Trends	Papers
Global software development	[25,37,42,45,48,53,63,65,67]
Implementation of software process technologies	[15,24,40,41,58,81,91]

(1) 分布式或者全球开发.在软件过程建模方法中,与分布式开发相关的一些最新研究主要包括:a) 提供一个可以支持分布式开发的支持环境,比如 Maurer 等人[65]提出了基于 COBRA 和 Java RMI 的分布式系统 MILOS,以支持 Internet 环境下的开发;Doppke 等人[67]将针对娱乐和游戏的虚拟环境(virtual environment)用于软件过程的建模和执行,使得软件过程可以由地理上分散的 Agents 来执行.b) 支持过程片断在不同组织间的传递和执行,比如 Li 等人[53]从移动性的角度重新审视了软件过程,并将软件过程的移动性映射到π演算,从而为其提供了一种简约和精确的描述方式;Wang[45]在 CAGIS 中提出了可以支持程序片断在不同工作区和地点间进行传递的框架,而 Becker 等人[42]通过代理(delegation)的方式将一部分执行过程打包,并交由合同商(contractor)执行,而发包方可以通过一些项目执行时的反馈信息对项目进行监控;Ge 等人[48]通过 Nets-within-net 也可以实现类似移动 Agent 的效果,从而支持过程片断的移动.c) 利用一些比较新的技术支持分布式开发,比如 Helland 等人[37]采用面向服务的体系架构(service oriented architecture),支持软件过程的分布式执行,而 Ruìz 等人[63]基于 MOF 和 XMI 提出了支持过程传递的方法.另外,Vanzin 等人[25]报告了在一个分布式环境中如何共同定义全球软件开发过程的经验.

(2) 软件过程技术实施.在软件过程建模技术实施过程中,软件过程建模语言的表示方式需要易于理解和应用[15,40],需要提供多种不同的视图[40]以及电子过程指导(electronic process guide,简称 EPG)[15,81],以利于相关人员更好地理解和执行软件过程.所采用的描述应该是说明性的(prescriptive or descriptive)[15,81],且具有比较好的结构(可以同时表示各种生命周期和具体开发活动的细化)[15];具有明确控制流的表达方式,与其他通过前、后置条件或者触发规则间接定义控制流的方式相比,更易于理解[41],但隐含描述控制流的方式具有较强的适应性和灵活性.另外,由于支持过程执行的需要,以及每个组织或者项目都有各自的信息,因此需要一个过程建模语言能够提供文本注释的方式,在过程模型中表示这些信息;但软件过程中额外的注释越多,则说明过程建模语言的表达能力与实际的需要相差就越远[41].

自然语言描述的软件过程比较易于理解,但自然语言表示的过程却难以进行最基本的一致性检查和其他分析[40].而根据软件过程实施的经验,必须在软件过程被应用之前确保过程本身的正确性,需要相应的过程建模语言能够支持形式化的分析[58].对过程进行分析的前提是,能够对软件过程必要的细节进行描述,同时包含过多的细节信息也因过程的抽象层次较低而失去了通用性,需要频繁地被更新[40].

另外,软件过程建模技术的实施离不开技术工具的支持,该工具不但需要对过程执行者(process performer)提供支持,而且还需要对过程工程师(process engineer)提供支持,也需要同时让过程执行者了解必要的概念,以及为什么要这么做[41,81].Becker-Kornstaedt 等人[40]还着重论述了在获取过程制定所必须的信息时的一些经验,von Wangenheim 等人[81]给出了在小规模企业中引入迭代开发过程的经验,而 Coleman[24]对于企业采用软件开发过程的经验报告也具有很好的参考意义.

通过表 5 和上面的分析,我们可以进一步得到如下结论:

软件开发过程的日益分布式和全球化是一个在不断增强的趋势.表 5 中关于这一趋势的研究,对于分布式

和全球化的支持进行了很多有益的探索,提出了一些理论方法和实现框架.但在上述研究的具体应用过程中,面临着如下的困难:相应的理论方法和实现框架的引入,需要每个组织内部能够很好地实现模型驱动的开发,能够准确地定义组织间的交互方式和接口,并可以处理由于涉及更多因素所带来的不可预测性,而这在现实的开发环境下往往很难被满足.

在实施过程中将会面临很多相互矛盾的需求,比如形式化和非形式化、抽象和具体、明确和非明确的控制流、易于理解和易于分析等.而其中的核心是软件过程建模方法的形式化需求和非形式化需求之间的矛盾:形式化和非形式化的区别,主要在于建模语言语义的严格程度.若一个建模方法被称作形式化的,则应该具有比较严格的操作语义.因此,设计一个新的过程建模语言的关键是如何在形式化和非形式化之间做到平衡[41],而一般的策略是针对过程的不同层次或者过程的不同部分,分别采用形式化程度不同的方法来表示[58,91].理想的情形是,不同程度的形式化方法之间的语义基本相同且可以实现相互间的转换,从而使得软件过程的表示和分析能够无缝地集成在一起[15].

4 一种多维度的集成化软件过程建模方法

随着软件开发过程复杂性、动态性和分布性的日益加强,客观上对于支持软件开发活动的 PSEE 提出了更高的要求,需要在软件过程的文档化、执行、分析、演化以及对于分布式的支持等方面,能够更好地满足由过程复杂性、动态性和分布性所带来的新需求.一个最为突出的问题就是由于软件开发活动涉及到了很多方面的要素,而如果只是从某些特定的方面去考虑和组织一个软件过程,将会因对某些内容的忽视而给开发项目带来各种各样的问题.为了更好地支持对软件开发中最为重要的过程、技术和经济要素的描述和管理,在综合了软件过程、软件工程经济学和软件开发技术这 3 个领域已有理论和方法的基础上,我们提出了集成化的软件过程框架 TRISO Model(TRidimensional integrated SOftware development model)[49,53].如图 8 所示,该模型的 3 个维度分别是软件工程过程维、软件工程技术维和软件工程人力维,而 3 个维度内的主要对象(first-class object)分别是活动(activity)、制品(artifact)和人员(actor).对于技术维,制品上的操作将会与开发技术密切相关(比如:代码重构涉及到重构技术、需求制定涉及需求分析和提取技术等),开发集成(development integration)主要从制品上所可以实施的行为,以及制品间的转换关系来描述软件工程中的技术因素.对于过程维,一系列有序的开发活动构成了该维的一个主线,其中的过程集成(process integration)主要描述活动间的并发和组合关系.对于人力维,主要是从人员所扮演的角色出发,每个角色具有与完成特定任务相关的一组行为以及一定的组织结构,服务集成(service integration)主要是描述人员之间完成特定任务的并发组合方式.三维之间的对象通过并发方式组合在一起,相互之间分别由数据集成(data integration)、使用集成(use integration)和管理集成(management integration)的方式结合在一起,共同构成了一个软件开发过程.图 9 给出了 3 个维度上对象间的交互模型,其中每个维度内的对象都可以采用顺序和并发的方式组合在一起,通过同步或者异步的方式传递信息,而维间对象通过并发的方式组合在一起,并通过同步方式实现相互间的协调和同步.

为了描述每个维度中的对象及其相互间的并发与交互,我们提出了具有严格操作语义的图形化建模语言 TRISO/ML(TRISO/modeling language)[47].基于 TRISO/ML 可以从行为角度,按照统一的方式描述软件过程中的角色、产品、技术、工具以及相互间的交互,减少从实际过程执行到虚拟过程执行的信息反馈过程中由人为因素所带来的不利影响;可以在更低的抽象层次和更细的粒度上描述开发活动、工作产品、角色以及其他要素,减少虚拟过程执行和实际过程执行因为描述力度不一致所带来的问题.在该语言中,每个对象具有一定的行为,且可以通过特定的方式与其他对象进行信息交互并实现同步.同时,由于其操作语义可以用多元π演算[102]精确地描述,因此,可以很好地支持软件过程的执行、分析(包括基于模型检测的软件过程正确性检查和过程间的匹配)、演化和分布式执行.图 10 给出了 TRISO/ML 的分析框架,首先用该语言的图形化符号描述给定的软件过程,而后根据转换规则可以自动地生成多元π演算的表达式,描述图形化过程的操作语义,而后在多元π演算的基础上可以支持软件过程的执行和一系列的分析.

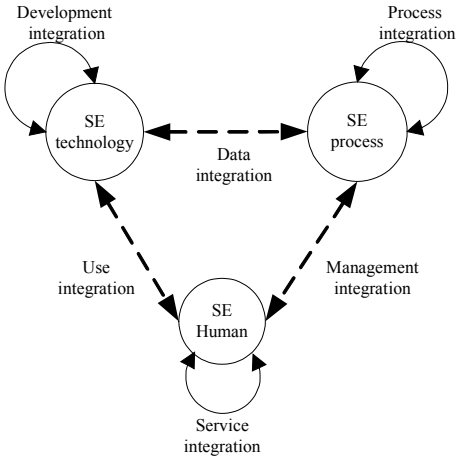

Fig.8 TRISO Model

图 8 TRISO Model 集成示意图[49]

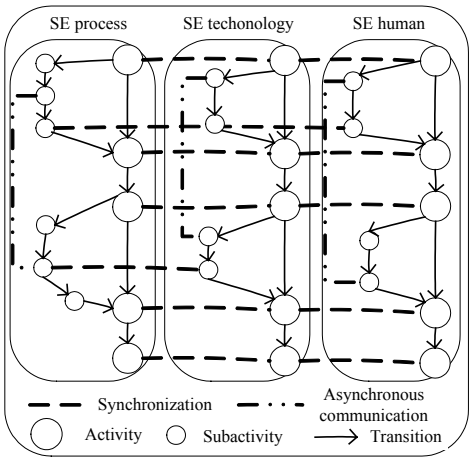

Fig.9 The dynamic semantics model of TRISO Model

图 9 TRISO Model 动态语义[53]

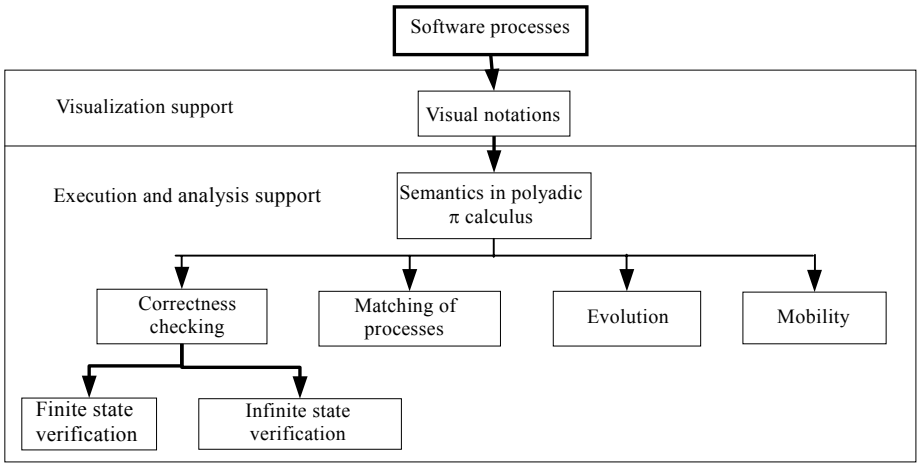

Fig.10 Framework for the representation and analysis of TRISO/ML

图 10 TRISO/ML 表示和分析框架

关于 TRISO/ML 的详细定义可参见文献[47],下面将分别讨论 TRISO/ML 如何支持软件过程的文档化、执行、分析、演化以及分布式执行.

文档化.TRISO/ML 采用面向行为(behavior oriented)的方式来描述软件过程,该语言提供了如图 11 所示的图形化符号用以描述行为间的并发、顺序和选择,基于通道的信息传递和同步.另外,可以通过注释(annotation)的方式表示其他信息.

执行.对于图形化的 TRISO/ML 过程,可以通过一系列的规则将其转换为多元π演算表达式.下面的规则说明了如何将一个包含活动和简单角色的 TRISO/ML 过程转换为多元π演算表达式:

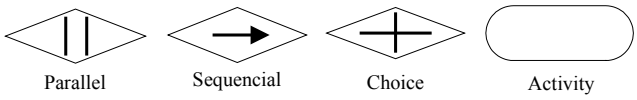

Fig.11 Notations of TRISO/ML

图 11 TRISO/ML 图形化表示符号

规则 1. 过程中人员的转换规则.假定一个角色或者执行人员用唯一的标识符 *ac* 来表示,则其对应的多元π演算表达式为

$$A_{ac} \overset{\text{def}}{=} assign_{ac}(start,end).\overline{start}.end.A_{ac} \mid A_{ac}\ .$$

上述规则表明:*ac* 的初始状态为空闲,在被分配任务时将得到与任务相关的两个通道 *start* 和 *end*,而所要执行的任务何时开始将取决于 *ac*,当其向 *start* 通道发送启动信号后,相应的活动将开始被执行,而在该任务完成后,*ac* 将从 *end* 通道接收到任务完成的信号,从而 *ac* 将再次恢复到空闲的状态.

而对于表示活动的节点,其π演算进程是由分别表示输入、输出和执行的 3 个子进程并发组成.若 *I* 表示输入子进程,*O* 表示输出子进程,*E* 表示执行子进程,则规则 2~规则 6 描述了如何表示 TRISO/ML 中的一个活动节点.

规则 2. 活动的转换规则.对于一个活动节点 *a*,若其从通道 $chi_1,\ldots,chi_l$ 分别接收到$\{b_{11},\ldots,b_{1m}\},\ldots,\{b_{l1},\ldots,b_{ln}\}$,从通道 $ex_{a_p\_a}$ 接收$(p_1,\ldots,p_u)$,然后通过通道 $cho_1,\ldots,cho_r$ 分别将 $\{c_{11},...,c_{1s}\}$, ...,$\{c_{r1},...,c_{rt}\}$ 发送出去,并且将 $\{r_1,...,r_v\}$ 通过通道 $ex_{a\_a_p}$ 返回给其父节点 a_p,则活动 *a* 的π-演算表达式为

$$A_a \overset{\text{def}}{=} (\nu\, i_1,...,i_l,i_o)(I_{a_s}\langle i_1,...,i_l\rangle \mid E_a\langle i_1,...,i_l,i_o\rangle \mid O_a\langle i_o\rangle)\ ,$$

$$I_a \overset{\text{def}}{=} (i_1,...,i_l).chi_1(b_{11},...,b_{1m}).\overline{i_1}\langle b_{11},...,b_{1m}\rangle \mid ... \mid chi_l(b_{l1},...,b_{ln}).\overline{i_l}\langle b_{l1},...,b_{ln}\rangle.ex_{a_p\_a}(p_1,...,p_v)\ ,$$

$$O_a \overset{\text{def}}{=} (i_o).i_o(c_{11},...,c_{1s},...,c_{r1},...,c_{rt},r_1,...,r_v).\overline{cho_1}\langle c_{r1},...,c_{rs}\rangle.....\overline{cho_r}\langle c_{r1},...,c_{rt}\rangle.\overline{ex_{a\_a_p}}\langle r_1,...,r_v\rangle\ ,$$

其中,执行子进程 E_a 与活动的类型相关,一个非终端活动可以细分为按如下时序组合的多个子活动:顺序活动、并行活动和选择活动,另外,终端活动也可以对其自身的 E_a 进行定义.规则 3~规则 6 分别讨论了这几种活动的 E_a 的转换细节.

规则 3. 顺序活动的表示.若 *a* 为一个非终端活动且其被拆分为 *w* 个顺序执行的子活动 $a_1,\ldots,a_w$,每个子活动都可能与父活动进行信息交换,比如第 *w* 个子活动从父活动 *a* 接收到$(p_{w1},\ldots,p_{wj})$并将$(r_{w1},\ldots,r_{wk})$返回给 *a*.同时,该活动被分配给人员 *ac*,则活动 *a* 的执行子进程可以描述为

$$\begin{aligned}
&E_a = (i_1,...,i_l,i_o).i_1(b_{11},...,b_{1m})...i_l(b_{l1},...,b_{ln}).ex_{a_p_a}(p_1,...,p_u).trigger_a.\overline{assign_{ac}}\langle start_a,end_a\rangle,\\
&start_a.\overline{ex_{a_a1}}\langle p_{11},...,p_{1h}\rangle.\overline{trigger_{a1}}.ex_{a1_a}(r_{11},...,r_{1i}).triggered_{a1}...\overline{ex_{a_aw}}\langle p_{w1},...,p_{wj}\rangle.\overline{trigger_{aw}},\\
&ex_{aw_a}(r_{w1},...,r_{wk}).triggered_{aw}.\overline{triggered_a}.\overline{end_a}.\overline{i_o}\langle c_{11},...,c_{1s},...,c_{r1},...,c_{rt},r_1,...,r_v\rangle.
\end{aligned}$$

在上面的规则中,由于活动 *a* 是非终端活动,执行进程 E_a 从进程 I_a 接受输入,然后按顺序关系触发 *a* 的子活动,执行完毕后将活动执行所得到的数据通过进程 O_a 输出.

规则 4. 并发活动的表示.当非终端节点 *a* 被拆分为 *t* 个并发的子活动时,则其 E_a 子进程表示为

$$\begin{aligned}
E_a &= (i,q,i_1,...,i_l,i_o).(\nu k_{a1},...,k_{aw})i_1(b_{11},...,b_{1m})...i_l(b_{l1},...,b_{ln}).ex_{a_p_a}(p_1,...,p_u).\\
&\quad trigger_a.\overline{assign_{ac}}\langle start_a,end_a\rangle.start_a.(E_1 \mid E_2),\\
E_1 &= (\overline{ex_{a_a1}}\langle p_{11},...,p_{1h}\rangle.\overline{trigger_{a1}}.ex_{a1_a}(r_{11},...,r_{1i}).triggered_{a1}.\overline{k_{a1}}.\overline{k_{a1}}\langle r_{11},...,r_{1i}\rangle) \mid ... \mid\\
&\quad (\overline{ex_{a_aw}}\langle p_{w1},...,p_{wj}\rangle.\overline{trigger_{aw}}.ex_{aw_a}(r_{w1},...,r_{wk}).triggered_{aw}.\overline{k_{aw}}.\overline{k_{aw}}\langle r_{w1},...,r_{wk}\rangle),\\
E_2 &= k_{a1}.k_{a1}(r_{11},...,r_{1i})...k_{aw}.k_{aw}(r_{w1},...,r_{wk}).\overline{triggered_a}.\overline{end_a}.\overline{i_o}\langle c_{11},...,c_{1s},...,c_{r1},...,c_{rt},r_1,...,r_v\rangle,
\end{aligned}$$

进程 E_1 触发执行所有的子活动,而 E_2 进程从各个子活动收集活动的执行结果并输出到进程 O_a.

规则 5. 选择活动的表示.若非终端活动 *a* 通过选择关系被拆分为 *t* 个子活动,则其 E_a 子进程被描述为

$$E_a = (i,q,i_1,...,i_l,i_o).(\nu k)i_1(b_{11},...,b_{1m})...i_l(b_{l1},...,b_{ln}).ex_{a_p\_a}(p_1,...,p_u).$$
$$trigger_a.\overline{assign_{ac}}\langle start_a,end_a\rangle.start_a.(E_1 \mid E_2),$$
$$E_1 = (\overline{ex_{a\_a1}}\langle p_1,...,p_h\rangle.\overline{trigger_{a1}}.ex_{a1\_a}(r_1,...,r_j).triggered_{a1}.\bar{k}.\bar{k}\langle r_1,...,r_j\rangle) + ... +$$
$$(\overline{ex_{a\_aw}}\langle p_1,...,p_h\rangle.\overline{trigger_{aw}}.ex_{aw\_a}(r_1,...,r_j).triggered_{aw}.\bar{k}.\bar{k}\langle r_1,...,r_j\rangle),$$
$$E_2 = k.k\langle r_1,...,r_j\rangle.\overline{triggered_a}.\overline{end_a}.\bar{i}_o\langle c_{11},...,c_{1s},...,c_{r1},...,c_{rt},r_1,...,r_v\rangle.$$

规则 6. 终端活动的表示.对于终端活动 a,其子进程 E_a 为

$$E_a = (i,q,i_1,...,i_l,i_o).i_1(x_{11},...,x_{1m})...i_l(x_{l1},...,x_{ln}).ex_{a_p\_a}(p_1,...,p_u).trigger_a.$$
$$\overline{assign_{ac}}\langle start_a,end_a\rangle.start_a.\overline{ex_{a\_ap}}\langle p_1,...,p_h\rangle.\overline{triggered_a}.\overline{end_a}\,\bar{i}_o\langle c_{11},...,c_{1s},...,c_{r1},...,c_{rt},r_1,...,r_v\rangle.$$

规则 7. 整个软件过程.整个软件过程被定义为所有活动和人员的并发组合:

$$SP = A_{a1} \mid ... \mid A_{am} \mid A_{ac1} \mid ... \mid A_{acn}.$$

为了分析或仿真一个软件过程,我们还需要一个辅助过程使得整个过程集合是一个封闭的系统,我们将其命名为 *Env*,它与过程 *SP* 并发组合.*Env* 可以定义为

$$Env = \overline{trigger_{root}}.triggered_{root},$$

其中,*root* 代表软件过程的根活动.

基于上面的规则,我们可以非常精确地描述一个 TRISO/ML 过程的行为,因此可以非常容易地支持过程的执行.

(1) 分析.通过自动转换得到的π演算表达式,可以从以下方面对所建模的过程进行分析:验证软件过程的正确性,主要是用时序逻辑表达的关于控制流和数据流的约束条件,我们已经使用有限状态验证工具,比如 CWB-NC[109]和 SPIN[110]验证μ演算表达的过程性质[47];过程间的匹配,由于进程代数特别是π演算定义了很多种具有不同严格程度的进程等价性,而且存在工具支持等价性的自动判断,比如 CWB-NC 和 MWB[111],因此这些方法和工具可以用于比较两个软件过程的一致性[47].

(2) 演化和分布式执行.在π演算中,用于进程间通信的通道既可以传递数据也可以传递通道本身,而通道实际上反映了系统对象之间的联系,因此可以通过通道在进程间的传递来描述具有动态结构的系统[112],而这种描述能力可以很好地支持软件过程的演化和分布式执行[51].

5 结 论

本文采用系统评价方法,对软件过程建模领域最近 10 年的研究给出了一个综合性的概括和分析.根据系统评价方法,首先明确本文所试图回答的 3 个问题,即软件过程建模方法的主要范式、主要研究目的和最新的发展趋势.其次,在明确问题的基础上根据特定的搜索策略和评价标准,选出了 72 篇文章作为该系统评价的证据.在分析和总结这些文章的基础上,对所提出的问题进行了系统的分析和论述:我们根据新的分类范式,对软件过程建模方法的研究进行了系统的分类,阐述了每种范式中的典型研究案例,对每种范式的优缺点及其所适用的场合进行了讨论;而按研究目的对软件过程建模方法进行的分类,有助于清晰地把握该领域研究所要解决的主要问题,以及与问题相对应的主要解决方案;对于分布式、全球软件开发的论述,反映了该领域对软件开发的国际化和全球化这一最新趋势,所做的主要研究工作,而对软件过程建模技术实施的讨论,总结了在实际开发环境中应用已有过程建模技术的最新经验.本文第 4 节所给出的多维度集成化软件过程建模方法可以通过一致的视图、灵活的抽象层次、准确地描述软件过程中的主要要素及其动态行为,使得过程模型与实际开发过程之间的粒度和抽象层次更趋于一致,有利于 PSEE 和实际开发环境更加紧密地集成.

References:

[1] Lonchamp J. A structured conceptual and terminological framework for software process engineering. In: Proc. of the ICSP. 1993. 41-53. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=236823&isnumber=6077

[2] Montangero C, Derniame JC, Kaba BA, Warboys B. The software process: Modelling and technology. In: Derniame JC, BAK, Wastell DG, eds. Proc. of the Software Process: Principles, Methodology, Technology. Springer-Verlag, 1999. 1−14.

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

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

[5] Dion R. Process improvement and the corporate balance sheet. IEEE Software, 1993,10(4):28−35.

[6] Kinnula A. Software Process Engineering Systems: Models And Industry Cases. ACTA Universitatis Ouluensis, 2001.

[7] Arbaoui S, Derniame J-C, Oquendo Fav, Verjus He. A comparative review of process-centered software engineering environments. Annal of Software Engineering, 2002,14(1-4):311−340.

[8] Gruhn V. Process-Centered software engineering environments. A brief history and future challenges. Annals Software Engineering, 2002,14(1-4):363−382.

[9] Osterweil LJ. Software processes are software too. In: Proc. of the 9th Int'l Conf. on Software Engineering (ICSE'87). IEEE Computer Society Press, 1987. 2−13.

[10] Osterweil LJ. Software processes are software too, revisited: An invited talk on the most influential paper of ICSE 9. In: Proc. of the 19th Int'l Conf. on Software Engineering (ICSE'97). ACM, 1997. 540−548.

[11] Conradi R, Jaccheri ML. Process modelling languages. In: Derniame JC, Wastell DG, eds. Proc. of the Software Process: Principles, Methodology, Technology. Springer-Verlag, 1999. 27−52.

[12] Zamli KZ, Lee PA. Taxonomy of process modeling languages. In: Proc. of the ACS/IEEE Int'l Conf. on Computer Systems and Applications (AICCSA 2001). IEEE Computer Society, 2001. 435−437.

[13] Zamli KZ, Isa NAM. A survey and analysis of process modeling languages. Malaysian Journal of Computer Science, 2004,17(2): 68−89.

[14] Dowson M, Fernstrom C. Towards requirements for enactment mechanisms. In: Proc. of the 3rd European Workshop on Software Process Technology (EWSPT'94). Springer-Verlag, 1994. 90−106.

[15] Kneuper R. Requirements on software process technology from the viewpoint of commercial software development: Recommendations for research directions. In: Proc. of the 6th European Workshop on Software Process Technology (EWSPT'98). Springer-Verlag, 1998. 111−115.

[16] Pohl K, Weidenhaupt K, Domges R, Haumer P, Jarke M, Klamma R. PRIME—Toward process-integrated modeling environments. ACM Trans. on Software Engineering and Methodology, 1999,8(4):343−410.

[17] Kitchenham B. Procedures for performing systematic reviews. Software Engineering Group, Department of Computer Science Keele University Keele, 2004.

[18] Pai M, McCulloch M, Gorman JD, Pai N, Enanoria W, Kennedy G, Tharyan P, Colford JMJ. Systematic reviews and meta-analyses: An illustrated, step-by-step guide. The National Medical Journal of India, 2004,17(2):86−95.

[19] Hannay JE. A systematic review of theory use in software engineering experiments. IEEE Trans. on Software Engineering, 2007,33(2):87−107.

[20] Kitchenham BA, Mendes E, Travassos GH. Cross versus within-company cost estimation studies: A systematic review. IEEE Trans. on Software Engineering, 2007,33(5):316−329.

[21] Jorgensen M, Shepperd M. A systematic review of software development cost estimation studies. IEEE Trans. on Software Engineering, 2007,33(1):33−53.

[22] Zhao X, Chan K, Li M. Applying agent technology to software process modeling and process-centered software engineering environment. In: Proc. of the 2005 ACM Symp. on Applied Computing (SAC 2005). ACM, 2005. 1529−1533.

[23] Ahmed-Nacer M. Towards a new approach on software process evolution. In: Proc. of the ACS/IEEE Int'l Conf. on Computer Systems and Applications (AICCSA 2001). IEEE Computer Society, 2001. 345−351.

[24] Coleman G. An empirical study of software process in practice. In: Proc. of the 38th Annual Hawaii Int'l Conf. on System Sciences (HICSS 2005)—Track 9. IEEE Computer Society, 2005. 315−320.

[25] Vanzin M, Ribeiro MB, Prikladnicki R, Ceccato I, Antunes D. Global software processes definition in a distributed environment. In: Proc. of the 29th Annual IEEE/NASA on Software Engineering Workshop (SEW 2005). IEEE Computer Society, 2005. 57−65.

[26] Padberg F. Linking software process modeling with markov decision theory. In: Proc. of the 28th Annual Int'l Computer Software and Applications Conf.—Workshops and Fast Abstracts (COMPSAC 2004). IEEE Computer Society, 2004. 152−155.

[27] Yoon IIC, Min SY, Bae DH. Tailoring and verifying software process. In: Proc. of the 8th Asia-Pacific on Software Engineering Conf. (APSEC 2001). IEEE Computer Society, 2001. 202−209.

[28] Lee S, Shim J, Wu C. A meta model approach using UML for task assignment policy in software process. In: Proc. of the 9th Asia-Pacific Software Engineering Conf. on APSEC 2002. IEEE Computer Society, 2002. 376−382.

[29] Padberg F. Using process simulation to compare scheduling strategies for software projects. In: Proc. of the 9th Asia-Pacific Software Engineering Conf. on APSEC 2002. IEEE Computer Society, 2002. 581−590.

[30] Ceravolo P, Damiani E, Marchesi M, Pinna S, Zavatarelli F. A ontology-based process modelling for XP. In: Proc. of the 10th Asia-Pacific Software Engineering Conf. on Software Engineering Conf. (APSEC 2003). IEEE Computer Society, 2003. 236−242.

[31] Atkinson DC, Weeks DC, Noll J. The design of evolutionary process modeling languages. In: Weeks DC, ed. Proc. of the Software Engineering Conf. on 2004 11th Asia-Pacific. Washington: IEEE Computer Society, 2004. 73−82.

[32] Huo M, Zhang H, Jeffery R. A systematic approach to process enactment analysis as input to software process improvement or tailoring. In: Proc. of the XIII Asia Pacific Software Engineering Conf. on APSEC 2006. IEEE Computer Society, 2006. 401−410.

[33] Bendraou R, Combemale B, Cregut X, Gervais MP. Definition of an Executable SPEM 2.0. In: Proc. of the 14th Asia-Pacific Software Engineering Conf. on APSEC 2007. IEEE Computer Society, 2007. 390−397.

[34] Meng X, Wang Y, Shi L, Wang F. A process pattern language for agile methods. In: Proc. of the 14th Asia-Pacific Software Engineering Conf. on APSEC 2007. IEEE Computer Society, 2007. 374−381.

[35] Park S, Choi K, Yoon K, Bae DH. Deriving software process simulation model from SPEM-based software process model. In: Choi K, ed. Proc. of the Asia-Pacific Software Engineering Conf. on APSEC 2007. 2007. 382−389.

[36] Yin R, Hu H, Ge J, Lu J. Quantitative analysis of value-based software processes using decision-based stochastic object Petri-nets. In: Proc. of the Asia-Pacific Software Engineering Conf. on APSEC 2007. 2007. 526−533.

[37] Helland T, Grundy J, Hosking J. A service-oriented architecture for software process technology. In: Proc. of the Australian Software Engineering Conf. (ASWEC 2006). IEEE Computer Society, 2006. 132−141.

[38] Franch X, Ribó JM. A structured approach to software process modelling. In: Proc. of the 24th Conf. on EUROMICRO. IEEE Computer Society, 1998. 753−762.

[39] Cares C, Franch X, Mayol E, Alvarez E. Goal-Driven agent-oriented software processes. In: Proc. of the 32nd EUROMICRO Conf. on Software Engineering and Advanced Applications (EUROMICRO 2006). IEEE Computer Society, 2006. 336−347.

[40] Becker-Kornstaedt U, Belau W. Descriptive process modeling in an industrial environment experience and guidelines. In: Proc. of the 7th European Workshop on Software Process Technology (EWSPT 2000). Springer-Verlag, 2000. 176−189.

[41] Becker-Kornstaedt U, Neu H, Hirche G. Software process technology transfer: Using a formal process notation to capture a software process in industry. In: Proc. of the 8th European Workshop on Software Process Technology (EWSPT 2001). Springer-Verlag, 2001. 63−76.

[42] Becker S, Jager D, Schleicher A, Westfechtel B. A delegation based model for distributed software process management. In: Proc. of the 8th European Workshop on Software Process Technology (EWSPT 2001). Springer-Verlag, 2001. 130−144.

[43] Balust JMR, Franch X. Building expressive and flexible process models using a UML-Based approach. In: Proc. of the 8th European Workshop on Software Process Technology (EWSPT 2001). Springer-Verlag, 2001. 152−172.

[44] Podnar I, Mikac B, Caric A. SDL based approach to software process modeling. In: Proc. of the 7th European Workshop on Software Process Technology (EWSPT 2000). Springer-Verlag, 2000. 190−202.

[45] Wang AI. Support for mobile software processes in CAGIS. In: Proc. of the 7th European Workshop on Software Process Technology (EWSPT 2000). Springer-Verlag, 2000. 115−130.

[46] Bhuta J, Boehm BW, Meyers S. Process elements: Components of software process architectures. In: Proc. of the Int'l Software Process Workshop (SPW 2005). LNCS 3840, Springer-Verlag, 2005. 332−346.

[47] Yang Q, Li M, Wang Q, Yang G, Zhai J, Li J, Hou L, Yang Y. An algebraic approach for managing inconsistencies in software processes. In: Proc. of the Int'l Conf. on Software Processes (ICSP 2007). LNCS 4470, Springer-Verlag, 2007. 121−133.

[48] Ge J, Hu H, Lu P, Hu H, Lü J. Translation of nets within nets in cross-organizational software process modeling. In: Proc. of the Int'l Software Process Workshop (SPW 2005). LNCS 3840, Springer-Verlag, 2005. 360−375.

[49] Li M. Expanding the horizons of software development processes: A 3-D integrated methodology. In: Proc. of the Int'l Software Process Workshop (SPW 2005). LNCS 3840, Springer-Verlag, 2005. 54−67.

[50] Chen B, Avrunin GS, Clarke LA, Osterweil LJ. Automatic fault tree derivation from Little-JIL process definitions. In: Proc. of the Int'l Software Process Workshop and Int'l Workshop on Software Process Simulation and Modeling (SPW/ProSim 2006). LNCS 3966, Springer-Verlag, 2006. 150−158.

[51] Li M, Yang Q, Zhai J, Yang G. On mobility of software processes. In: Proc. of the Int'l Software Process Workshop and Int'l Workshop on Software Process Simulation and Modeling (SPW/ProSim 2006). LNCS 3966, Springer-Verlag, 2006. 105−114.

[52] Wang Q, Xiao J, Li M, Nisar MW, Yuan R, Zhang L. A process-agent construction method for software process modeling in SoftPM. In: Proc. of the Int'l Software Process Workshop and Int'l Workshop on Software Process Simulation and Modeling (SPW/ProSim 2006). LNCS 3966, Springer-Verlag, 2006. 204−213.

[53] Li M. Assessing 3-D integrated software development processes: A new benchmark. In: Proc. of the Int'l Software Process Workshop and Int'l Workshop on Software Process Simulation and Modeling (SPW/ProSim 2006). LNCS 3966, Springer-Verlag, 2006. 15−38.

[54] Gary K, Lindquist T, Koehnemann H, Derniame J. Component-Based software process support. In: Proc. of the 13th IEEE Int'l Conf. on Automated Software Engineering (ASE 1998). IEEE Computer Society, 1998. 196−199.

[55] Bendraou R, Gervais MP, Blanc X. UML4SPM: An executable software process modeling language providing high-level abstractions. In: Proc. of the 10th IEEE Int'l Enterprise Distributed Object Computing Conf. (EDOC 2006). IEEE Computer Society, 2006. 297−306.

[56] Reis CAL, Reis RQ, Abreu M, Schlebbe H, Nunes DJ. Flexible software process enactment support in the APSEE model. In: Proc. of the IEEE 2002 Symp. on Human Centric Computing Languages and Environments (HCC 2002). IEEE Computer Society, 2002. 112−121.

[57] Acuna ST, Sosa MDV. An integral software process formal model based on the SOCCA approach. In: Sosa MDV, ed. Proc. of the XX Int'l Conf. of the Chilean Computer Science Society (SCCC 2000). 2000. 162−171.

[58] Gruhn V, Urbainczyk J. Software process modeling and enactment: An experience report related to problem tracking in an industrial project. In: Proc. of the 20th Int'l Conf. on Software Engineering (ICSE 1998). IEEE Computer Society, 1998. 13−21.

[59] Cass AG, Lerner BS, Sutton JSM, McCall EK, Wise A, Osterweil LJ. Little-JIL/Juliette: A process definition language and interpreter. In: Proc. of the 22nd Int'l Conf. on Software Engineering (ICSE 2000). ACM, 2000. 754−757.

[60] Ge J, Hu H, Gu Q, Lu J. Modeling multi-view software process with object petri nets. In: Proc. of the Int'l Conf. on Software Engineering Advances (ICSEA 2007). IEEE Computer Society, 2006. 41−46.

[61] Tran HN, Coulette B, Dong BT. Modeling process patterns and their application. In: Proc. of the Int'l Conf. on Software Engineering Advances (ICSEA 2007). IEEE Computer Society, 2007. 15−20.

[62] Baldassarre MT, Caivano D, Visaggio CA, Visaggio G. ProMisE: A framework for process models customization to the operative context. In: Proc. of the 2002 Int'l Symp. on Empirical Software Engineering (ISESE 2002). IEEE Computer Society, 2002. 103−110.

[63] Francisco R, Aurora V, Felix G, Piattini M. Using XMI and MOF for representation and interchange of software processes. In: Proc. of the 14th Int'l Workshop on Database and Expert Systems Applications (DEXA 2003). IEEE Computer Society, 2003. 739−744.

[64] Franch X, Ribo JM. Using UML for modelling the static part of a software process. In: Proc. of the 2nd Int'l Conf. on Unified Modeling Language Beyond the Standard (UML 1999). Springer-Verlag, 1723, 1999. 292−307.

[65] Maurer F, Dellen B. Internet based software process support. In: Proc. of the 7th Workshop on Enabling Technologies (WETICE 1998). IEEE Computer Society, 1998. 27−32.

[66] Cook JE, Wolf AL. Software process validation: Quantitatively measuring the correspondence of a process to a model. ACM Trans. on Software Engineering Methodology, 1999,8(2):147−176.

[67] Doppke JC, Heimbigner D, Wolf AL. Software process modeling and execution within virtual environments. ACM Trans. on Software Engineering Methodology, 1998,7(1):1−40.

[68] Jaccheri ML, Picco GP, Lago P. Eliciting software process models with the E3 language. ACM Trans. on Software Engineering Methodology, 1998,7(4):368−410.

[69] Dami S, Estublier J, Amiour M. APEL: A graphical yet executable formalism for process modeling. Automated Software Engineering, 1998,5(1):61−96.

[70] Deiters W, Gruhn V. Process management in practice applying the FUNSOFT net approach to large-scale processes. Automated Software Engg., 1998,5(1):7−25.

[71] Min SY, Lee HD, Bae DH. SoftPM: A software process management system reconciling formalism with easiness. Information and Software Technology, 2000,42(1):1−16.

[72] Aalst WMPvd. Formalization and verification of event-driven process chains. Information and Software Technology, 1999,41(10): 639−650.

[73] Atkinson DC, Weeks DC, Noll J. Tool support for iterative software process modeling. Information and Software Technology, 2007,49(5):493−514.

[74] Breu R, Huber W, Schwerin W. A conformity model of software processes. Information and Software Technology, 2001,43(5): 339−349.

[75] Chou SC, Chen JJY. Process evolution support in concurrent software process language environment. Information and Software Technology, 1999,41(8):507−524.

[76] Fuggetta A, Jaccheri ML. Dynamic partitioning of complex process models. Information and Software Technology, 2000,42(4): 281−291.

[77] Henderson-Sellers B, Gonzalez-Perez C. A comparison of four process metamodels and the creation of a new generic standard. Information and Software Technology, 2005,47(1):49−65.

[78] Odeh M, Kamm R. Bridging the gap between business models and system models. Information and Software Technology, 2003, 45(15):1053−1060.

[79] Smith H. Business process management-the third wave: Business process modelling language (bpml) and its pi-calculus foundations. Information and Software Technology, 2003,45(15):1065−1069.

[80] Wallace C. Using Alloy in process modelling. Information and Software Technology, 2003,45(15):1031−1043.

[81] Wangenheim CGv, Weber S, Hauck JCR, Silva GTd. Experiences on establishing software processes in small companies. Information and Software Technology, 2006,48(9):890−900.

[82] Canfora G, GarcíaFel, Piattini M, Ruiz F, Visaggio CA. A family of experiments to validate metrics for software process models. Journal of Systems and Software, 2005,77(2):113−129.

[83] Donzelli P, Iazeolla G. Hybrid simulation modelling of the software process. Journal of Systems and Software, 2001,59(3): 227−235.

[84] Kellner MI, Madachy RJ, Raffo D. Software process simulation modeling: Why? What? How? Journal of Systems and Software, 1999,46(2-3):91−105.

[85] Pfahl D, Lebsanft K. Integration of system dynamics modelling with descriptive process modelling and goal-oriented measurement. Journal of Systems and Software, 1999,46(2-3):135−150.

[86] Cobleigh JM, Clark LA, Osterweil LJ. Verifying properties of process definitions. SIGSOFT Software Engineering Notes, 2000, 25(5):96−101.

[87] Estublier J, Amiour M, Dami S. Building a federation of process support systems. SIGSOFT Software Engineering Notes, 1999, 24(2):197−206.

[88] Dirk J, Schleicher A, Westfechtel B. Using UML for software process modeling. SIGSOFT Software Engineering Notes, 1999, 24(6):91−108.

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

[90] Donzelli P, Iazeolla G. A hybrid software process simulation model. Software Process: Improvement and Practice, 2001,6(2):97−109.

[91] Barros MdO, Werner CaML, Travassos GH. A system dynamics metamodel for software process modeling. Software Process: Improvement and Practice, 2002,7(3-4):161−172.

[92] Object Management G. Unified Modeling Language (UMLircledR). 2007. http://www.omg.org/spec/UML/Current

[93] Object Management G. Software process engineering metamodel (SPEM). 2005. http://www.omg.org/technology/documents/formal/spem.htm

[94] Object Management G. Meta object facility (MOF). 2006. http://www.omg.org/spec/MOF/2.0/

[95] Object Management G. Ontology definition metamodel (ODM). 2007. http://www.omg.org/cgi-bin/doc?ptc/2007-09-09

[96] Chen PP-S. The entity-relationship model—Toward a unified view of data. ACM Trans. on Database System, 1976,1(1):9−36.

[97] Costin Badica, Amelia Badica, Litoiu V. Role activity diagrams as finite state processes. In: Proc. of the 2nd Int'l Symp. on Parallel and Distributed Computing (ISPDC 2003). IEEE Computer Society, 2003. 15−22.

[98] Heimann P, Joeris G, Krapp CA, Westfechtel B. DYNAMITE: Dynamic task nets for software process management. In: ICSE '96: Proc. of the 18th Int'l Conf. on Software engineering. IEEE Computer Society, 1996. 331−341.

[99] Int'l Telecommunication U. Series Z: Lanuages and General Software Apects for Telecommunication Systems-Formal Description Techniques (FDT)-Specification and Description Language (SDL). 1999. http://www.itu.int/ITU-T/studygroups/com10/languages/Z.100_1199.pdf

[100] Valk Rud. Object Petri nets: Using the nets-within-nets paradigm. In: Lectures on Concurrency and Petri Nets. Springer-Verlag, 2003. 819−848.

[101] Hoare CAR. Communicating Sequential Processes. Upper Saddle River: Prentice-Hall, Inc., 1985.

[102] Sangiorgi D, Walker D. PI-Calculus: A Theory of Mobile Processes. New York: Cambridge University Press, 2001.

[103] Zambonelli F, Omicini A. Challenges and research directions in agent-oriented software engineering. Autonomous Agents and Multi-Agent Systems, 2004,9(3):253−283.

[104] Belkhatir N, Estublier J, Melo Wel. ADELE-TEMPO: An environment to support process modelling and enaction. In: Software Process Modelling and Technology. Taunton: Research Studies Press Ltd., 1994.187−222.

[105] Cassandras CG, Lafortune S. Introduction to Discrete Event Systems. Secaucus: Springer-Verlag, Inc., 2006.

[106] Kirkwood CW. System dynamic models: A quick introduction. 1998. http://www.public.asu.edu/~kirkwood/sysdyn/SDIntro/SDIntro.htm

[107] Herbsleb JD, Moitra D. Guest editors' introduction: Global software development. IEEE Software, 2001,18(2):16−20.

[108] Souza CRBD. Global software development: Challenges and Perspectives. http://citeseer.ist.psu.edu/457465.html, 2002.

[109] Cleaveland R, Li T, Sims S. The concurrency workbench of the new century: User's manual. SUNY at Stony Brook, 2000.

[110] Holzmann GJ. The model checker SPIN. IEEE Trans. on Software Engineering, 1997,23(5):279−295.

[111] Bor V. A verification tool for the polyadic π-calculus. Department of Computer Systems, Uppsala University, 1994.

[112] Milner R, Joachim P, Walker D. A calculus of mobile processes-part I and II. Journal of Information and Computation, 1992, 100:1−77.

李明树(1966－),男,吉林德惠人,博士,研究员,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.

杨秋松(1977－),男,博士生,助理研究员,主要研究领域为软件过程方法与技术.

翟健(1981－),男,博士生,主要研究领域为软件过程方法与技术.