计算机科学 1997 Vol. 24№. 4

软件过程及其集成环境的研究

Studies of the Software Process and Its Integrated Environment

郭 江 黄 涛

（中国科学院软件研究所 北京 100080）

摘 要 软件过程对软件开发的进度、费用和质量都有很大的影响，因此正日益受到学术界和工业界的关注，并开发了一些软件过程的集成环境以提供相应的支持。此外，软件过程模型和元模型对过程的改进都起着关键的作用。本文对以上问题都进行了探讨。

关键词 软件过程，过程环境，过程模型

1. 研究软件过程的意义

软件技术同计算机硬件一样发展了几十年，但软件生产的水平还远远不能满足社会的需求。在计算机出现的最初十几年（60年代以前），软件的设计基本上是机器指令或汇编语言的艺术。60年代起，高级语言的广泛使用促进了计算机在各个领域的应用，但是随着软件系统越来越复杂，在60年代末产生了“软件危机”。这时的程序设计还不是一门科学或技术而是一门“艺术”，软件设计基本上没有什么方法和准则可以遵循。对于小的软件系统，开发方法还不显得重要，但随着应用领域的不断拓广，硬件规模的日益庞大，结构也越来越复杂，导致软件的规模也不断增大，复杂程度和成本也越来越高，一个软件系统往往要几十、上百、上千个人年才能完成。如IBM公司的OS/360系统耗费了几千万美元，花费了五千多人年。在60年代末出现“软件危机”的情况下，提出了软件工程的概念，并在较短的时间内发展成为了一个完整的学科方向。在软件工程学科方向形成以来的20多年间，在理论研究和工程实践两个方向进行了大量的工作，多种方法和技术的研究取得了长足的进展，其中有些已经比较成熟，并广泛地应用于软件开发之中，也有一些新方法和新技术还在不断的探索之中。

程序设计方法研究的是小规模程序设计，而软件开发方法则是研究大规模软件的开发过程。软件过程是研究在软件开发过程中如何组织、管理人员和资源，指导人们开发软件系统的学科。

从70年代开始，人们陆续提出了一些软件开发方法，如结构化方法，与之相应的是瀑布模型，将整个周期划分为需求分析、系统设计、编码、测试与软件维护几个阶段，提出了软件生命周期的概念，成为开发软件产品的一个行之有效的工程化模型。瀑布模型经过多年的发展和完善，已成为人们广泛接受的一种传统和标准的开发模式。

从80年代开始，人们逐渐认识到瀑布模型的不足，为了开发出更可靠、执行效率更高、功能更完善的软件，就需要软件工程研究人员不断开发新的软件开发模型。此后提出了其它开发模型，如快速抛弃原型法，增量式开发法、自动软件综合法、空间螺旋式开发模型以及面向对象的开发技术。它们都从不同的侧重点出发，试图避免瀑布模型的缺陷，描述适合软件开发规律的模式。

但是困扰人们的“软件危机”并没有得到很好解决，大规模软件生产仍然很困难。据Helen E. Thomson估计[1]，目前每年大约有15%的软件项目被取消；而对于大规模的软件开发项目而言，大约有50%的项目不仅超出预算，而且不能按时完成[2]。就是小规模的项目，开发情况也不容乐观，据KPMG的统计[3]，几乎所有小项目都超出预算20%，而且延期20%。可见大多数软件开发企业的软件开发情况并不理想，软件开发的进度、质量以及成本得不到有效的控制。而软件工程的主要目的是提高软件产品的质量和生产率、缩短软件系统的开发周期、降低

成本。这就迫使软件工程的研究人员去寻找改善软件开发状况的另一种途径。

在软件开发过程中影响产品的质量、进度和成本的因素很多，其中主要的可控因素有：开发人员的技能和经验，所使用的技术和工具，产品的类型和复杂程度，以及开发的总体环境（如进度的压力和通讯情况）[4]。不同类型的开发有不同的生产率，如新产品的开发、产品的改进、产品的移植和产品的维护等均不相同。因此一些研究人员就转向了软件开发过程的研究。

尽管软件过程只是改进软件质量和组织性能的几个可控因素之一，如图1所示。但是M. Dowson指出："软件产品的质量在很大程度上依赖于软件过程，尤其是大规模的软件开发更是如此"[5]。

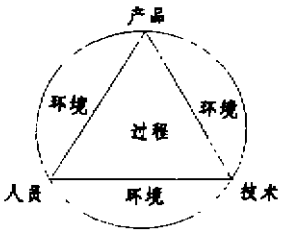

图1 影响产品的主要可控因素

因此，不少软件开发企业力图通过改进开发过程来改善软件产品的质量、提高软件生产率、缩短产品的开发时间，从而增加企业的竞争力和收益。但是过程改进是一个很复杂的问题，也不能一蹴而就。于是采用了许多方法，如配置管理、功能点分析、软件质量保证、可靠性工程以及全面质量管理等。

企业软件过程的改进通常是商业竞争，软件定购或增加利润的结果。企业在评价了它目前的开发情况和过程成熟度之后，就可以改进企业软件的开发进程了。主要包括的几个方面是改进方法的选择、改进方法的实现，以及改进的度量。改进的选择和能否成功实现依赖于许多因素，如当前企业的过程成熟度、开发人员的基本技能、成本、风险及实现速度。预测某个过程的改进能否成功也是很困难的，这涉及许多其它因素：人员技能、可接受性、培训的有效性以及实现的效率。

软件过程改进所带来的效益是巨大的，如Schlumberger公司在这方面就取得了很大的成功[6]。该公司是一个分布在100多个国家的跨国公司，雇员有来自75个国家的五万三千人，产品包括石油、电子、仪表、以及计算机辅助设计和生产等等。十多年前，软件在其产品中只占很少一部分，但是现在占了很多产品工程投资的50%-100%，为此Schlumberger的高级管理部门开始了软件过程的改进工作。于1989年成立了Schlumberger计算机科学实验室（SLCS），进行提高软件产品质量和生产率的研究。SLCS分布在两个地方，美国Texas州的Austin和法国的Montrouge。Schlumberger的工程组有大有小，从5个人到180个人都有，所用的机器从4位的微处理器到超级并行计算机。SLCS对软件过程的改进基于美国CMU的SEI的CMM（能力成熟模型），这种技术是用一个多阶段的方法来促进过程的改进。SLCS在1989年进行了初步研究，确定了开发软件产品的76个组织，这76个组织中的雇员总数达2000多人。在1990年初对其中的20个进行了过程改进，和200人进行了面谈，这20个组织的雇员占Schlumberger软件开发人员总数的70%，随后将过程改进扩大到三十个组织。

Schlumberger的软件过程改进是成功的。1990年中期开始进行过程改进的一个工程组，其1990年工程项目按时完成率为51%，1991年为89%，1992年为94%。两年半以前，另一个组的工程项目的完成时间通常是计划时间的两倍，即计划期内只完成进度的50%。在改进了项目规划和跟踪之后（增加了项目状态报告和按期评审），在1991年计划期内完成进度的87%，而在1992年则基本上是按原计划完成了。同时尽管产品的规模有了增加，但产品的质量还是有了改进。在过程改进开始时，总共40万行无注释源代码（NCSS）的产品的出错率为0.22/千行。现在70万行无注释源代码的产品的出错率仅为0.13/千行。在质量保证方面，Schlumberger在1990年初成立了SQA（软件质量保证）小组。在1989年，Schlumberger的软件产品有25%在交付之后被用户发现了错误，而在SQA小组制定了交付目标以后，到1990年底，只有10%的产品在交付之后被用户发现了错误。

2. 能力成熟模型（CMM）的研究现状

美国卡内基·梅隆大学（CMU）的软件工程研究所（SEI）在软件过程方面作出了巨大的贡献。1986年11月，SEI在Mitre公司的帮助下，以IBM的工

作为基础开发了一个过程成熟框架，用于帮助企业改进软件过程。这项工作最初是为了给美国政府提供一个方法来评价其软件供货商的能力[7]。在1987年9月，W. Humphrey 和 W. Sweet 交付了一个过程成熟框架的简要描述和一个成熟问题表[8]。SEI 希望这个成熟问题表能提供一个简单的手段来确定企业的软件过程需要改进的领域。但通常人们认为这个问题表就是"模型"而不是一个表达过程成熟性问题的调查表。M. Kellner 和 G. Hansen 于 1988 年在报告[9]中提出了软件过程建模的概念，其目标是将新的软件工程技术引入实际工作中。Kellner 还在该报告中讨论了 SEI 进行的软件过程建模实例的研究。

W. Humphrey 和 M. Kellner 于 1989 年在报告[10]中指出，为了完成企业的任务和改善企业的工作方式，需要用一个过程框架给企业提供一个定义好的软件过程。用于建模的总体框架不仅简化了开发过程模型的任务，而且还可以根据个人的需要进行裁剪。该报告还描述实体过程模型的建模方法。同年，D. Kitson 和 W. Humphrey 在报告[11]中阐述了评价在改进企业软件能力方面所扮演的角色，尤其是在调动企业的开发能力，降低成本、加快进度和提高质量等方面所起的作用。D. Kitson 和 W. Humphrey 在报告中还从概念和程序的角度描述了软件过程评价，并对软件过程、过程管理和过程成熟的基本概念进行了讨论，这些综合起来就构成了一个软件过程评价和改进的框架。在这一年，W. Humphrey 在报告[12]中指出过程的自动化既增加了其优点，也扩大了其缺点。自动化可以使一个有效的过程更有效，也能使一个混乱的过程更糟糕，如果公司还没有确定软件过程改进计划，那么尽管花了大价钱购买了 CASE 工具，但还是解决不了问题。该报告还讨论了软件过程成熟框架及其和规划与安装 CASE 系统的关系，指出过程不是一个包治百病的神丹妙药，由于 CASE 系统可能需要大量的投资，因此必须考虑成本和收益问题。

在软件过程成熟框架和问题表使用了四年之的，SEI 积累了很多经验，M. Paulk 等终于在 1991 提交了报告[13]，将软件过程成熟框架开发成一个定义完整的模型。系统而有原则地使用这个模型就能产生出一个成熟问题表。通过全面应用该成熟框架，就可以给企业提供一个创建过程改进程序更有效的方法(比如成熟问题表)。通过使用从软件过程评价和来自工业与政府的大量返馈知识，就产生了过程成熟框架的改进版本，称为软件的能力成熟模型(Capability Maturity Model，CMM)，这是 CMM Version 1.0。CMM 包括三个基本方面：一个五层的软件开发能力模型、一个企业软件开发实践问题表、一个根据问题表建立企业成熟度的"算法"。五层模型是全面质量管理的五个层次的软件解释。属于第一层(最低层)的软件开发企业在管理问题上而非技术问题上正处于初级阶段，管理问题包括项目管理、质量控制、变化控制。但是这不意味着第一层的企业开发不出成功的应用系统，它只是说明企业开发软件系统的过程没有很好定义，或者在应用中有了问题。属于第二层的软件开发企业虽然有了基本的管理控制，但是截止期、成本和用户需求的满足性等企业性能依赖于项目组中具体的开发人员。一个有过开发同种应用系统类型经验的小组在这方面就会很好，而一个没有经验的小组在这方面就会很差。在第三层的软件开发企业中，由于每个开发人员都使用相同的开发过程，因此企业性能一致性不依赖于项目组中具体的开发人员。在这一层上，并不要求开发人员对所有的项目均遵循一样的开发过程，而是要根据每个项目的具体情况，在过程构造块的基础上形成最适合实际要求的过程。在剩下的两个较高层上，企业拥有了度量程序(第四层)，并对收集的度量数据进行分析、对过程进行优化(第五层)。

根据 CMM，W. S. Humphrey 在 1989 年对企业评价的结果[14]是：85%的企业处于第一层，14%的处于第二层，1%的处于第三层。两年半之后，J. Baumert 对美国的 296 个项目进行了评价[15]，结果表明，尽管仍然存在着很大的管理问题，但情况已经有了改善，81%的企业处于第一层，12%的处于第二层，7%的处于第三层。

M. Paulk 等在 1993 年推出了 CMM Software Version 1.1[16]。在 CMM 的基础之上，E. Averill 等人于同年推出了软件能力评价(SCE)[17]。SCE 提供了一种评价企业软件过程能力的手段，即对企业管理生产软件过程的好坏进行评价。此外，SCE 还提供了一种方法来比较软件供货商的软件开发能力和预定义的标准之间的差距。

R. Park 等于 1994 年在报告[18]中指出，过程改进的初衷就是为了正确地对软件费用和进度进行估计。在 CMM 的激励和鼓舞下，他们确定了在开发可重复性软件过程中进行估价和费用管理的关键。这个报告的最初内容包括模板、标准和指南，用于开发定义的估价过程、培训材料、估价实践的例子，以及现代费用模型满足估价需求的能力评价。这一年 T. Olson 和 N. Reizer 在报告[19]中提出了软件过程框架(SPF)，使用一种表格方式，给 SEI 开发的 CMM 提供有关的信息，这样就更适用于过程的定义和改进。SPF 帮助用户确定企业的软件过程文档是否和 CMM 推荐的保持一致。当发现不一致时，SPF 提供了支持决策的能力，这样就有助于 CMM 的应用。

S. Masters 和 C. Bothwell 于 1995 年在报告 CMM Appraisal Framework (CAF)[20] 中描述了基于 CMM 的评价方法。这种方法是正在开发的 CBA (CMM-Based Appraisal)项目的一部分。CAF 提供了一个用于衡量一个企业的过程成熟度(与 CMM 相对应)框架。它包括了一个基于 CMM 的评价方法和一个总的评价体系结构，并定义了开发基于 CAF 的评价方法的需求。随后，SEI 在报告[21]中提供了评价企业软件估价过程和资源的能力标准的问题表，用以支持好的估价过程。问题表可用于收集信息，支持过程评价并指导企业进行过程改进方面的工作。

3. 软件过程集成环境的研究进展

由于软件工程界认识到了软件过程的重要性，因此不仅提交了很多报告，还开发了许多支持软件过程的大型项目。主要有美国 N. Goldman 主持的 AP5[22]，S. Sutton 和 L. Osterweil 主持的 APPL/A[23]，M. Karr 主持的 E-L[24]，K. Huff 主持的 GRAPPLE[25]，G. Kaiser 主持的 MSL/Marvel[26]，P. Feiler 主持的 SADT，M. Kellner 主持的 STATEMATE[27]；意威 R. Conradi 主持的 EPOS/OOER[28]；日本 M. Suzuki 主持的 HFSP[29]；德国 W. Schafer 主持的 MERLIN；法国 J-C Derniame 主持的 MASP/DL，M. Bourdon 主持的 OPIUM，N. Belkhatir 主持的 Adele[30]；欧洲软件研究所 G. Koch 主持的 BOOTSTRAP[31]等等。下面介绍一些较新的研究项目。

台湾交通大学的 Chen Jen-Yen 和 Tu Chia-Ming 于 1994 年在 SUN 工作站上开发了一个并发软件过程语言(CSPL)[32]，用作一个以过程为中心的环境。和过程程序设计语言一样，CSPL 过程语言也分为语法和语义两个方面。由于 Ada 描述复杂软件的能力较强，因而 CSPL 的语法大多数继承了 Ada 的特征，这类似于另一个有名的过程建模语言 APPL/A，是对传统 Ada 语言的扩充。源于过程运作的目的，CSPL 的语义来自 UNIX Shell，CSPL 过程程序可以直接译成 UNIX Shell 来运行软件过程，这和 Process Weaver 差不多。Process Weaver 使用了 Unix Co-Shell 来运行软件过程，Unix Co-Shell 类似于通常的 Shell，但它可以移植到其它系统中。除了语法和语义之外，CSPL 还提供了一个环境来支持指导、模拟和运作软件过程。

所有软件项目的开发总会有一些不可避免的修改，这就会影响到原定的目标、决策和进度。这样，管理人员就需要获取有效可靠的数据来改进过程和产品。从这个角度出发，BARI 大学的 G. Visaggio 于 1994 年主持开发了一个称为 Prometheus[34]的产品，Prometheus 集成了过程的建模、过程的执行和管理、以及过程和产品有关数据的获取，它还帮助管理人员收集与某个特定项目有关的事件(包括预料的和没有预料的)数据，并提供了一个框架对这些事件进行分类，记录事件对过程和产品的影响。Prometheus 主要由三个基本部分组成：过程建模支持环境、过程模型转换工具(即将其转换成可执行的计划)、存储有关过程元素的数据仓库管理系统。Prometheus 主要支持软件开发过程中的以下几个方面：功能(什么活动产生什么产品)、组织(谁完成活动)、管理(何时、何地、以及为什么要进行这些活动)、行为(活动怎样完成)，前两条是由过程模型来描述，后两条则构成了执行规划。Prometheus 的主要目标是，通过过程的形式化描述给用户提供灵活、有效的过程执行规划以及过程模型和执行规划间的关系。在此基础上，Prometheus 帮助用户收集产品和过程的度量数据，这些度量数据是项目执行过程中的一个组成部分，包括产品数量、活动完成人员以及项目中断时间等等。

SPADE 是由 Carlo Ghezzi 主持于 1993 年在 Politecnico di Milano 开发的一个环境[35]，其主要目的是支持软件过程的分析、设计和运作。SPADE 的

核心是一个称为 SLANG(Spade LANGuage)的语言，它是描述软件过程建模和运作的一个特定语言。SLANG 基于时序的高层 Petri 网形式化机制，称为 ER 网。ER 网可以描述并发和实时系统，SLANG 的语言结构可以编译成 ER 网来运行。这样 SLANG 就提供了活动的执行模型，以及活动执行的不同选择方式。在 SPADE 中，过程和活动的状态主要用 ER 网标记来描述，事件的产生用 ER 网的状态转换表示，这样活动的执行就可以采用相应于活动的开始和结束事件来驱动。SPADE 的过程模型扮演着可执行"代码"的角色，因此过程的执行实际上是由过程模型解释器 SLANG Interpreter 来完成。SPADE 是用 C^{++} 和面向对象的数据库 O_2 开发的。

除了上面介绍的大型研究项目以外，随着学术界和工业界对软件开发过程越来越重视，不断有新的过程管理环境出现。据 David Sharon 和 Rodney Bell 在 1995 的统计[36]，近一、二年推出的过程管理方面的环境主要包括以下几方面的功能：①建模，用于项目的建模并将之映射到由工具支持的任务上；②工具协调，用于将工具的使用协调起来以便支持过程中的各个任务；③运作，用于指导和支持项目的有关人员遵循开发过程进行各项活动；④管理，用于帮助项目管理人员确定开发活动对完成过程的影响，根据实际情况做出相应的决策。这些工具的主要代表有：

(1)In Concert：是 Xsoft 公司的产品。该产品是一个工作流管理软件，用于描述和协调工作过程的主要有关部分—人员、活动以及信息。InConcert 将面向对象的工作流技术和文档管理服务集成在一起，并使用了一个基于任务的模型来协调过程。管理人员可以使用一个 GUI 来定义、修改和跟踪工作流过程。开发人员可以根据图形环境获得所需的文档，并使用一个 API(应用程序接口)和软件集成工具开发应用系统。

(2)Process Engineer：是 Learmonth & Burchett Management Systems 公司的产品。该产品是一个交互式的项目管理环境，用于支持开发过程的控制和自动化，其目标是对 SEI 的 CMM 提供支持。该环境可以根据软件开发企业和项目的具体情况进行裁剪，并可以在窗口环境中和其它工具集成在一起。Process Engineer 由一个可配置的过程库和访问及管理过程库所需的软件组成。这个过程库包括过程模板、可重用的过程核以及一个超文本的过程指南。Process Engineer 还提供了一个可视化的过程建模工具、一个专家级的过程分析器以及一个可配置的过程度量驱动工具。Process Engineer 可以和该公司的另一个产品 Project Manager 工具一起使用。

(3)Process Weaver：是 CAP 公司的产品。该产品包括几个部分：①Agenda 用于给每个用户提供一个要完成的当前任务集，用户可以使用与任务相关的工具处理任务的输入和输出对象；②Method & Activity Editor 用于帮助用户描述企业的软件开发方法学；③Process & Work Editor 用于帮助用户描述活动的工作顺序和组织情况。

(4)SE Companion：是 SECA 公司的产品。该产品包括一个多媒体、超文本的方法学(指南)、一个集成的分析和设计培训组件以及一个支持环境。作为一个完整的过程集成环境，用户可以调用 CASE 工具、项目管理工具、字处理器以及 SE Companion 中的其它部件。在 Help 工具、用户注释和用户文档中增加了声音交互功能，这样就便于有关人员在应用系统的开发过程中进行沟通，记录项目管理人员对开发人员的指示。由于该系统是完全面向用户的，因而项目管理人员可以根据自己的具体情况重新组织方法学，也可以修改指南本身。SE Companion 有三个版本，一个和 Intersolv 公司的 Excelerator 兼容、一个和 KnowledegWare 公司的 Application Development Workbench 兼容、一个不针对任何特定的 CASE 工具。

(5)Synergy：是 CASE Methods Development 公司的产品。该产品是一个基于 GUI 的系统，是 CASE/Framework 的一部分。该公司的开发方法学不仅支持各种项目的开发，从战略信息规划到客户/服务器应用系统等等，而且还支持对现存系统的改进。开发人员可以根据实际情况定义方法学和过程模板，以满足项目的特殊需求。Synergy 可以跟踪和管理不断变化的需求、有关问题的阐述、以及质量检查的报告。Synergy 的群组软件特征使之更容易搜集和报告项目信息，而且还有一个集成接口可以和其它许多项目管理系统相连接，如 Microsoft 公司的 Project、ABT 公司的 Project Workbench。在此基础上，Synergy 提供了对 CASE 工具的直接访问功能，

如 Texas Instrument 公司的 Information Engineering Facility，KnowledgeWare 公司的 Application Development Workbench。

(6)SynerVision：是 Hewlett-Packard 公司的产品。该产品帮助用户构造计算机辅助过程管理环境。一旦过程定义好以后，过程管理环境就给任务完成人员提供实时通讯以及所使用工具状态的报告。SynerVision 包括一些通用的模板，称为可编程的过程蓝图。用户可以对蓝图进行编程，根据具体信息将过程分成多个任务，把信息添加到每个任务的描述之中，包括软件工具间所需的通讯、任务的优先级、相关标准。用户还可以确定每个阶段必须收集的度量信息，并将之对应到状态报告中。

4. 软件过程模型

本文所讨论的过程模型，就是真实世界中的过程及其描述(即模型)，并重点强调元过程对过程模型的支持。在本文中，过程模型就是过程活动模型；只是过程模型更强调整个项目的过程，而过程活动模型更强调活动。开发过程模型的活动很复杂、也很关键，通常把用于描述其它活动的过程称为元过程。过程实际上就是活动的偏序集，与之相关的是产品、工具和资源，用于产生所需的输出；而元过程是一个元活动集，用于对一个过程进行建模、分析和支持，即过程模型是元过程处理的对象[37]。

一个过程模型及其形式化必须考虑实际过程的需求和特点[38]，主要的相关特征包括：①模块化，如使用层次结构子模型或可重用的模型对过程进行分段。②抽象，即使用逐步确定条件的方法形成特定的应用。③针对用户裁剪，一个有用的模型必须不断演化以反映企业的变化并不断进行改进，而且它还必须能针对特定的用户进行裁剪。④形式化，用于支持过程的自动分析和评估。⑤监控和反馈工具，用于帮助实现过程的评估和演化。⑥明确性和正交性，以便可以随意结合进较小的概念集，而不影响过程的整体情况。⑦人的可理解性，这可以通过外部图形表示来实现，在本文中使用可视化语言(VPML)来增加过程模型的可理解性。

过程活动的形式化机制一般可以概括为五类：①基于触发器、规则或描述性的形式化机制(ADELE2[30])；②基于网络的形式化机制，如扩充的 Petri 网(SPADE[35])；③解释性的程序设计形式化机制(IPSE2.5[39])；④混合形式化机制(EPOS[38])；⑤可以执行的可视化语言形式化机制(VPML[40])。

最初，可能只存在一个非正式的过程模型、该过程模型的需求以及高层设计，这个过程需求和设计可以逐步发展成更正式、更普遍的过程模型，硯个普遍而抽象的模型又可以不断提精细化，并针对用户进行改进，于是就形成一个更特殊的过程模型，这样就产生了一个具体的过程，可进行分析和模拟，最后产生可执行的过程模型，生产有关的产品。这里包含两类技术：①应用支持技术：是在过程中使用的特定领域的计算机工具等。②过程支持技术：是支持过程本身以及过程的方法、原则、语言、形式化以及工具的技术，这包括描述过程建模和实例化工具、过程分析和模拟工具以及过程运作和监控工具。

开发过程模型首先要对外部过程模型进行提取和抽象，产生一个非正式的过程模型，包括执行过程所接受到的反馈[41]。随后对过程的活动顺序以及所涉及的产品进行抽象描述，它可用于许多相似的项目和组织之中，共享某些信息。这样的模式意味着是一个非常高层的过程体系结构，可以满足一般的原则和需求。在软件生产中，这些模式称为软件生命周期，如瀑布模型和空间模型。在此基础上，用户可以根据具体情况详细而又严格地描述过程中的各种任务类型，这时可以从管理的角度来提精细化所有的子模型，如对特定领域的工具和角色进行分类和施加约束。在将软件模块交付集成测试组时，可以说明这是某个特殊过程模型所产生的结果。同时这个过程模型必须进一步实例化以适应项目的各种约束和条件，因而实例化的过程模型是一个可执行的过程描述，它将实例化的活动(即任务类型实例化成具体的任务)与具体的产品和项目资源联系起来，并遵循一定的调度时间表。一旦过程模型为相应的项目进行了实例化，该模型就可以用于模拟和运作。在过程模拟期间，需要工具来帮助分析、评价和模拟过程模型，并得到许多反馈信息，包括资源使用情况、资源瓶颈、活动完成时间、人员等待时间等等。有了这些反馈信息就可以对实例化过程进行修改，使之趋于合理化。在模拟之后就可以对过程模型进行实际运作，生产有关的产品。

5. 元过程模型

每个过程首先需要确定范围并进行抽象，然后针对用户进行修改、实例化、分析、模拟和改进，最后实施生产，因而就需要元过程及其元子过程来描述过程模型连续不断的改进，过程及其元过程的关系如图 2 所示。其中：①过程和元过程是由人来操纵的，他可以扮演不同的角色，如过程建立者、过程管理者等；②过程的输出之一是操纵人员对处理和工具使用情况的反馈，这些反馈驱动着元过程活动，使过程本身不断得到改进；③元过程不断产生和修改过程模型。为了实现这一目标，元过程使用相关的应用支持技术(AST)描述(模型)，这在很大程度上依赖于各种元过程工具。工具是过程支持技术(PST)的一部分，通常可分成两类：过程工具，用以支持过程执行人员(在过程实施阶段)；元过程工具，用以支持过程的建模活动；④元过程的输出之一是反馈给应用支持技术(AST)的评价和需求，这些主要涉及到特定领域工具的性能和效率。图 2 中的元过程进一步提精细化就得到图 3，其中有两个重要的元活动：①过程模型需求与总体设计的开发；②过程模型与支持环境的开发，即产生可执行的过程模型并实施，同时进行过程技术开发(PTD)，如图 4 所示。

这里有两个重要的问题要考虑：①增量式定义：必须能够以增量的方式生成、实例化和执行过程模型。在某些情况下，若干过程部分及其模型直到执行时才确定，因而就造成滞后链接或动态链接。增量式定义带来了极大的灵活性，有助于避免过程的不匹配。②反馈循环：过程模型必须在过程激活的情况下也能修改，即在模型实例化和执行之后仍能修改，这是为了适应模型所描述的企业及技术支持的变化，这些变化必须能迅速响应，以便有效地管理企业。当然，这是非常困难的一点。

正如上所述，产生过程模型的元活动是由人来操纵的，这些人可以扮演不同的角色[38]：①过程所有者：不仅负责提供过程描述、对过程性能进行评价，而且还负责过程的管理，这样就可以对过程改进提供额外的反馈和需求。②过程模型设计者：负责过程的设计、理解过程的本质、抽取和创建非正式的过程模型。随后建立具有一般性的过程模型，并将该过程模型应用到具体情况下。③过程管理者：负责提供信息并针对用户确定和实例化过程模型，而且还负责模拟和监控该实例化模型的执行。④过程执行者：负责过程模型的执行，并提供反馈信息，他是在企业过程中负责操纵的人员，在运作中他可以使用应用领域的规则、工具以及技术，这些均受过程运作环境的支持和控制。⑤过程技术提供者：支持所有过程的建模、模拟和运作，是过程技术开发(PTD)活动的负责人。

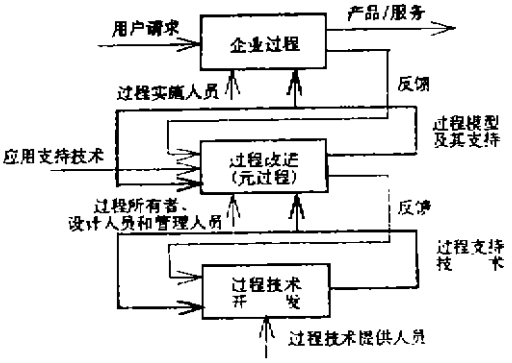

图 2 过程及元过程

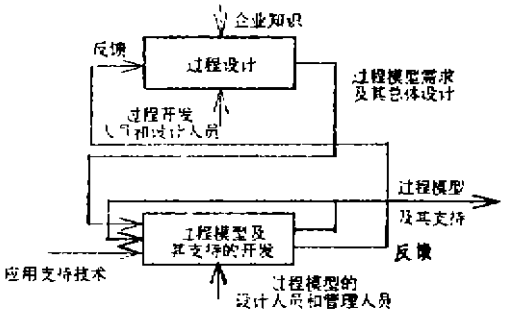

图 3 过程的改进(元过程)

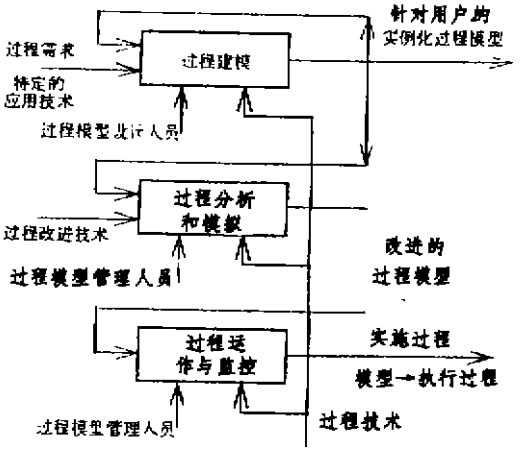

图 4 过程模型及支持的开发

6. 软件过程模型的概念模式

研究过程模型的一个很重要方面是确定组成软件开发过程的概念模式信息，这些信息属于许多不同的类型，如图5所示。

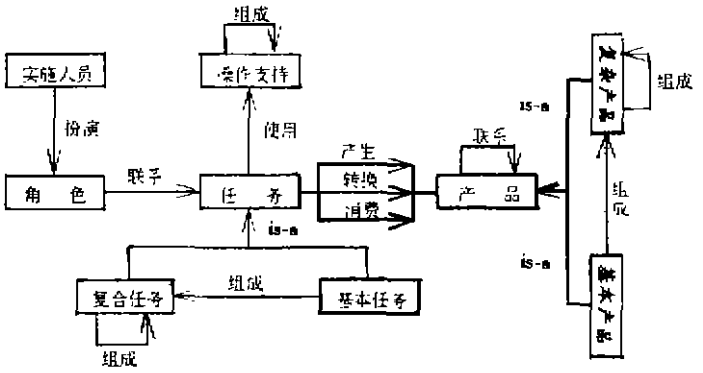

图5 过程模型的概念模式

一个软件过程是由任务(活动)集组成的，这些任务又可提精细化，分成若干子任务，这样一直下去，直到分解成基本任务为止。一般而言，高层任务相应于管理任务。在任务执行时，就会创建、消耗和修改产品。

产品表示了真实世界对象，是基本成份的聚集。在软件过程执行期间，产品的行为是由状态集来描述的。由于任务的开始依赖于工具使用情况及其产品的当前状态，因而这些产品的结构形成了处理产品基本成份的任务执行顺序。

在执行给定的任务时需要人力和物质资源，执行者代表负责任务管理的人或工具，他可以由能力集来描述。操作支持可以是在任务执行期间使用的一个工具集，也可以是用其知识来完成任务的开发人员。有两类基本任务：一类是简单任务，由单个人来执行，如编写一个函数；另一类是与多个开发人员密切相关的任务，这是由多个人(小组)来执行的，如会面碰头等。

每个任务至少与一个角色联系在一起，这个角色是在任务实施期间由一个或多个执行者扮演的。一个角色负责一个任务集，为此可以区别各种不同的角色，负责一项任务的管理人员、顾问、以及参加者等等。一个任务的管理人员负责任务的执行及所产生的产品质量。顾问是拥有完成任务所需知识的专家，参加者是实现任务产品的人员，一个角色可以与几个任务联系起来。

结束语 随着学术界和工业界对软件过程的研究和开发的关注，不仅提交了许多研究报告，而且还开发了一些支持软件过程的集成环境。作者对近年来各国同行在软件过程方面的工作进行了追踪、分析、研究和总结，并在此基础上提出了软件过程管理的系统模型 M-POID[42]，以 M-POID 为方法学开发了一个软件过程的集成环境 ISPE[43]。目前，正是软件过程的研究成果向产品转化的时期，不久就会有更多的软件过程支持产品出现，这将不仅有助于缩短软件产品的开发周期，而且还将提高软件产品的质量。

(参考文献共43篇略)

(上接第29页)

参考文献

[1] P. Beckman, Portable Parallel Programming in HPC++, In ICPP'96 Proc.

[2] A. S. Girmshaw, Easy-to-use Object-oriented Parallel Processing with Mentat, Computer, 26(5)1993

[3] P. A. G. Silivotti, A Tutorial for CC++, Technical Report CS-TR-94-02, California Institute of Technology, 1994

[4] J. Bannning, A Method for Determining the Side Effect of Procedure Calls, Ph. D. thesis, Stanford University, 1978

[5] M. Burke, etc., Interproceduaral Dependence Analysis and Parallelization, ACM SIGPLAN'86 Symposium on Compiler Conf. 12(7)1986

[6] M. B. Girkar, Functional Parallelism: Theoretical Foundation And Implementation, Ph. D. dissertation, University of Illinois at Urbana-Champaign, 1991

[7] Z. Li & P. Yew, Interprocedural Analysis for Parallel Programs, In Proc. Int. Conf. on Parallel Processing, 1988

[8] G. Zhu, L. Xie and Z. Sun, A New Mehtodology of Data Dependence Analysis for Parallelizing C++, ACM Sigplan Notice, 30(12)1995

[9] G. Zhu, L. Xie and Z. Sun, PRG: Procedure Reference Analysis in Extracting Non-Data Parallelism, Proc. of ICA3PP'96, 1996