软件工程环境 开放式系统 框架 (13)

计算机科学1994Vol.21№.6

58-62

第三代软件工程环境研究

——一个可配置的开放式系统框架

张 莉

(北京航空航天大学软件工程研究所 北京100083)

摘 要 Now, many works have been done for the next generation of software engineering environment. Software process modeling is introduced into the software engineering environment, and by instantiated and enacted, the process model can drive software engineering environment running. Furthermore, by substituting, evolving the software process model, the software engineering environment can be reconfigurated for a special project or organization. In this paper, the software engineering environment framework of the third generation will be introduced first. Then for example, its enactment and the process model supports in ALF will be discussed.

关键词 Software engineering environment (SEE), CASE environment, Software process model, Process integration.

关于软件工程环境有多种定义。E. Fedchak曾将其定义为"共同构成软件开发与支持框架的一组工具、结构、规则和方法(procedures)的集合。"[2]。C. J. Tully 在第九届国际软件工程年会(ICSE9)上又将软件工程环境定义为"为了获得更高的生产率和更高的产品质量,用于支持程序设计者、软件工程师、系统设计者和项目管理者等活动的一组计算机辅助设施。"[1]这两个定义从不同的角度,反映了软件工程环境的组成和意图。在此基础上,新一代软件工程环境又引入了新的概念。

第一代软件工程环境是一些工具的松散集合,用于支持软件过程的不同阶段。其典型例子是 UNIX 工具箱。第二代软件工程环境则是通过一个公共对象管理系统和/或一个公共用户接口系统将这些工具集成起来,形成一个集成环境。其典型例子是集成的计算机辅助软件工程环境(CASEE)、集成的项目支持环境或集成的程序设计支持环境(IPSE)。至此,CASEE 和 IPSE 便成了软件工程环境的代名词。

无论是第一代还是第二代软件工程环境,都包含了软件过程模型和软件工程环境两个含义,也就是说每个环境的实现都是针对某一种软件过程模型,如瀑布模型、快速原型、螺旋模型等。这样,第一、第二代环境就限制了环境中可执行的软件过程的特征,并通过提供一组按一定顺序执行的工具来实现这个过程模型。因此存在第一、第二代环境和软件过程模型两个概念。那么怎样才能开发出一个新的环境,既能适应某些特殊需求,又能通过裁剪适应不同项目和不同过程模型的需求呢?这就是本文将要描述的第三代软件工程环境。

1. 第三代软件工程环境框架:一个可配置的开放式系统

作为第三代软件工程环境,其特点在于软件过程模型不是固定的。这就是说可以将一软件过程模型(可替换地)放入第三代环境中,环境根据软件过程模型提供的信息运行。要实现这种灵活性,必须提供软件过程模型的解释执行机制,以保证软件环境能根据所提供的软件过程模型在运行中执行相应的软件过程。图1是 W. Schafer 等提出的第三代软件工程环境的一个典型框架。位于顶层的控制组件表示解释执行机制。SPM 表示软件过程模型(Software_Process_Model),决定软件工程环境的实际运行行为。中间 $T_1,\cdots,T_n$ 表示工具。下层基本构件是在第二代软件工程环境中已用过的用户接口系统(UIS)和对象管理系统(OMS)。箭头表示构件间的使用关系。

张莉 博士生,导师周伯生教授,主攻方向:软件工程环境和软件生产自动化、软件过程模型。

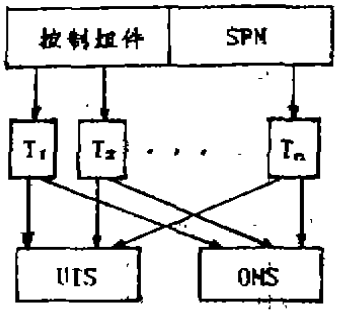

图1 第三代软件工程环境体系结构1

作者认为这个图虽然引入了过程模型但并不能完全反映第三代软件工程环境的思想，为此给出图2。其中，OMS 作为中心信息仓库，支持数据的集成；而工具插件之上的过程管理设施则根据指定的过程模型选择工具，配置支持某一特定项目过程的软件工程环境。进一步与烤面包模型比较，发现可以看作

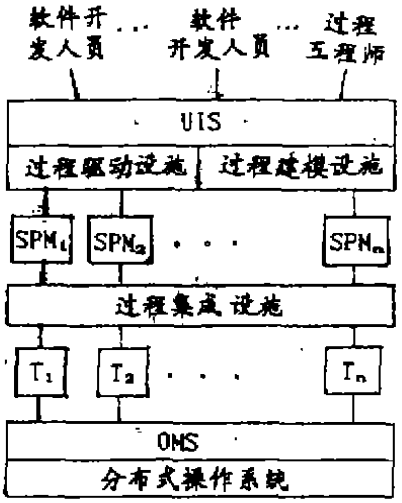

图2 第三代软件工程环境体系结构2

是它的一个扩充，见图3、图4。

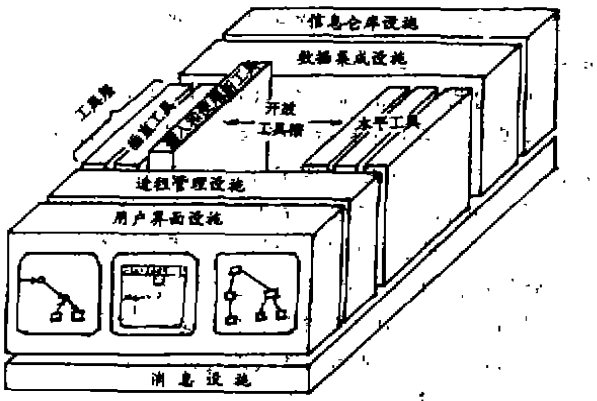

图3 NIST/ECMA 烤面包模型

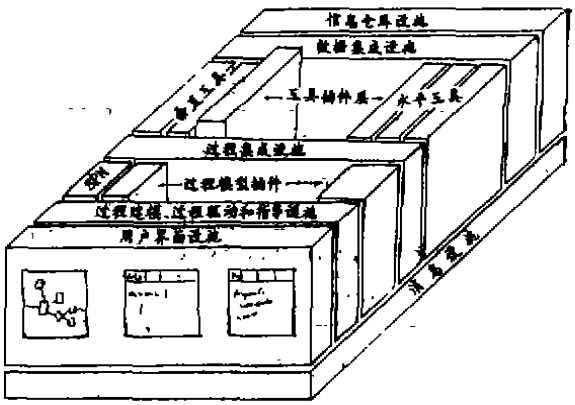

图4 改进的烤面包模型

图4引入了过程插件的思想，其中插入的过程可以是一个实例化的软件过程模型，也可以是通用过程模型，在系统中进行实例化；甚至可以利用系统所提供的建模工具建立自己的过程模型或修改已有的模型。因此二者的主要区别在于第二代软件工程环境仍以 OMS 和 UIS 为核心集成各阶段工具以支持某些软件开发过程模型（如瀑布模型，快速原型等，部分系统有一定的任务管理设施），而第三代软件工程环境则引入了可运作的过程模型的概念，以过程模型为中心驱动环境的工作。因此在第二代软件工程环境中，对软件过程的支持是被动的；而在第三代软件工程环境中，软件过程则是主动的因素。而且，我们可以看到，在第三代软件工程环境中，通过替换不同的软件过程模型，以及同一软件过程模型，用不同参数将其实例化，可以在第三代软件工程环境中配置不同的软件支持环境（对于分布式系统，可以同时支持多个项目过程）。又由于第二代软件工程环境所提供的对象管理系统 OMS 是一个开放式的 CASE 仓库（如 PCTE，CAIS），因此环境中可加入新的工具进行扩充，或替换已有工具。为此，我们说第三代软件工程环境是一个可配置的开放式系统，同时也是一个软件过程驱动的通用软件工程环境。

目前，着手于开发第三代软件工程环境的项目有：ESPRIT 计划中的 ALF 项目[4,5,14]、ESF 项目、

Arcadia 项目[16]和 ISTAR 项目[17]等。虽尚无一个成熟的系统，但仍在一定程度上体现了第三代软件工程环境的思想。下面我们主要借助于 ALF 项目，来讨论第三代软件工程环境的实现机制。

2. 第三代软件工程环境的实现机制

要实现这样一个软件过程驱动的软件工程环境，除了要有第二代软件工程环境作为基础外，还必须提供：(1)过程表达，即提供软件过程建模的形式化方法和建模支持。(2)实例化机制，根据所给项目实例化过程模型，提供可运作的过程模型。(3)环境生成器，产生能运作已实例化的软件工程模型的软件工程环境。

下面在介绍 ALF 总体结构的基础上，对这三部分进行讨论。

2.1 ALF 项目总体结构

ALF 项目的实现主要包括两个部分[6]：一是软件过程模型的开发环境，见图5。在 ALF 项目中称为 ALF 系统，是项目的核心工作；二是过程的运作环境，即基于 ALF 的集成项目支持环境(IPSE)，见图6。

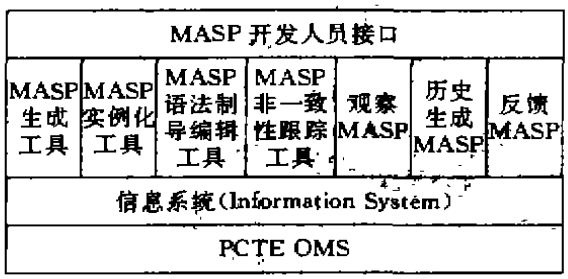

图5 ALF 系统体系结构

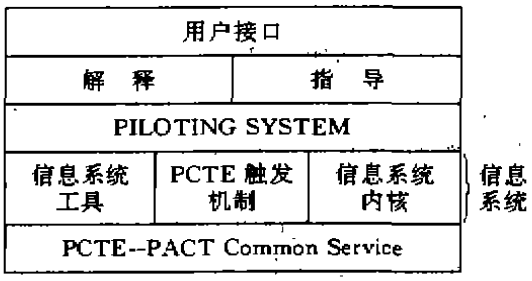

图6 基于 ALF 的 IPSE 体系结构

ALF 系统主要支持过程工程师的工作。语法制导过程编辑器、非一致性跟踪工具为开发语法、语义正确的过程模型提供了保证；过程模型实例化工具支持过程模型的实例化，产生可运作的过程模型；数据收集工具、历史过程库以及反馈机制支持过程的改进。ALF 的 IPSE 则根据所提供的实例化过程模型，完成过程的实际运作，指导最终用户(系统设计人员、程序设计人员、项目管理人员等)的活动。

2.2 软件过程模型

M. I. Kellner 在第四届软件过程研讨会上指出，成功的软件过程建模应包括以下四个方面：

(1)能有效地支持软件开发人员、管理人员、过程开发人员、质量保证人员之间的过程通讯。

(2)支持软件过程模型的重用。

(3)支持软件过程模型的演进。

(4)支持软件过程的管理。

为满足这四个要求，ALF 开发了一种形式化方法，称为 MASP (*Model for Assisted Software Process*)，为严格描述计算机辅助软件过程模型提供了一种方法。由于该描述是非二义性的，因此有助于对软件开发过程的进一步理解，和开发过程中各人员间的过程通讯。

MASP 概念为通用软件过程模型的描述提供了支持。所谓通用软件过程模型，是指可被增量地或重复地实例化，为特定的项目或组织生成专用软件过程模型。从这个意义上讲，软件过程模型可被方便地重用。

软件过程模型的演进在 MASP 中是通过对软件过程模型的实例化和运作交替进行来支持的。它允许软件过程模型在"运作中"的修改，包括组织结构的修改、项目决策的修改等，但所有这些修改都要通过调用环境所提供的反馈机制，来调整正在运作中的软件过程模型，从而完成对过程动态性的支持。

最后，MASP 中对软件过程的管理提供了相应的机制，包括规则推理和调度软件过程各种活动，控制其运作，监控其进展情况等。

MASP 所描述的辅助软件过程模型，包括一组对象模型，建立在 PCTE 的 OMS 之上，在 MASP 运作中定义 MASP 可操作的对象库；一组操作符类型，定义在对象类型之上，描述各类操作符的语义；和一组控制模型，控制操作符类型的运作，它包括用于过程状态描述的(逻辑)表达式、用于描述可接受操作符序列的排序(Orderings)，以及用于描述完整性约束和自动冲突处理的特征和规则。关于其语文定义和进一步描述，可参见文[4、5、14]。

2.3 实例化软件过程模型

由于软件过程模型是软件过程的抽象描述，在运作前，必须加入具体的项目和组织信息，即实例化。实例化的软件过程模型是软件过程的静态描述，它包括了其运作所需要的所有信息。

在 ALF 中，MASP 的实例化包括以下几个组件的实例化：

·对象模型的实例化：指在 IMASP（实例 MASP）运作之前确定对象，包括在 MASP 实例化过程明确提供对象，或作为操作符实例化的输入参数。

·操作符类型的实例化：MASP 是一个分层结构，由许多复杂的操作符类型组成。实例化时，其中底层最基本的操作符类型必须与满足其语义要求的工具相连；而高层被细化的操作符类型的实例化，则通过细化这些操作符类型的 MASP 的实例化来完成。

·其它，包括表达式、排序（ordering）、规则和特征的实例化只需用实际值或值地址替换表达式、排序、规则等中出现的形式参数。

考虑软件过程模型实例化的方法之一是在运作前将整个过程模型实例化，然后运作已得到的实例化过程模型，但正如 Lehman 在 ICSE9 年会上指出的[3]，这种工作方法——即"静态"实例化方法不够灵活，这是因为一个软件过程可能要持续很长时间，几周、几个月甚至几年，因此没有理由在运作开始前就要求一个完整的实例。静态实例化的另一个缺点是没有机会根据实际情况调整软件过程行为，因为每个活动、操作在运作前就定死了，不可能对当前软件过程模型的结果作出反应。因此，必须考虑过程的"动态"方面，考虑"运作中"的动态实例化。

为此，ALF 中提出"lazy_instantiation"（懒惰实例化）的概念，也就是说过程模型的实例化和运作交替进行，将操作符类型和对象类型的实例化推后，只有在需要实例化时才进行各个相应对象类型和操作符类型的实例化。这样在进行每步实例化时，都可考虑已执行了的软件过程，这就比静态实例化提供了更大的灵活性。

2.4 生成软件工程环境和过程的运作

所谓生成软件工程环境，这里指的是配置一个工作环境来运作实例化的软件过程，以辅助软件生产或使生产过程自动化。每个工作环境视其中完成的任务不同，由不同的资源和活动组成。这样的工作环境作为一个整体就是一个较小但复杂的软件工程环境，它有自己的用户界面。工作环境的概念是文[12,13]中提出的。在 OPM（Object Process Modeling）中，每个过程都有自己的窗口，在每个窗口上，显示和监视所有可用的资源和资源上的活动。

在 ALF 中，过程的运作是由 Piloting 系统支持的（见图6）。

当一个软件过程模型被实例化时，所得到的实例化软件过程模型（如 IMASP）便带有足够的信息，为运作这个实例化的软件过程模型配置相应的工作环境。生成的软件工程环境的主要软件包括：

（1）对应于 IMASP 对象模型的对象库管理工具，由 PCTE 的 OMS 和信息系统（包括附加的触发机制）支持，保证过程信息的存储和检索，以及对象库的完整性和并发性控制。

（2）支持 IMASP 操作符类型的一组工具，这里是由建立在 PCTE 上的第二代环境 PACT 提供的。

（3）IMASP 解释器（MINT），解释执行 IMASP，由 Piloting 系统支持。在 MASP 中，过程模型的运作实际是将 IMASP 作为辅助软件过程的一个局域知识库，对 MASP 的一个解释，在系统的开发过程中指导或辅助系统的开发。

IMASP 解释器（MINT）是一个软件工具，建立在 PCTE 机制和一个产生式系统 ALFRete（基于 Rete 匹配算法）[4]基础上，我们可将其看作一个"专家服务器"，因为该工具可在控制下的软件过程活动上进行向前、向后推理，具有推理功能，同时还可解释与 IMASP 相连的多个辅助软件过程（ASP），并协调其间的动态信息流。在实际工作中，ASP 可看成这些服务器的"客户"或一个专用 IMASP 的"用户"，不同 ASP 是同一静态文本实例化后的不同动态文本。过程中，有许多可能的活动要进行，这些活动由规则、操作符的前置和后置条件、调用序列控制以及用户要求来控制，并在必要的时候为用户提供帮助。

小结

以上主要讨论了第三代软件工程环境的框架，以及运作机制。目前，对于过程驱动的软件工程环境的研究，仍持有不同的态度。1992年9月在 Colorado 的 Boulder 举行的 Process_sensitive Software Engineering Environment Architecture 研讨会上所提出的几种意见便是个典型的代表[9]。如 Coggins，Wasserman 认为用软件工程环境来强制过程是注定要失败的，而 Heimbigner 则认为在近期内要得到一个完全集成的系统是不可能的等等。综合各家意见，结合自己的看法，我认为过程控制向自动化方向发展是正确的，而在近期内得到一个完全集成的自动化系统也是不可能的，因为首先第三代软件工程环境要建立在第二代软件工程环境成熟的基础上。它以过程为中心，配置所需的工具，甚至支持在过程运作中工具的插入或拆卸。因此，它要求工具插件的思想得到真正的实现，这要求环境通用接口（包括用户界面）完全的标准化，并要求足够的标准化 CASE 工

具作为后盾。其次，对过程模型本身的研究尚不成熟。因此，目前的工作一方面集中于对过程的理解，研究过程的体系结构、过程建模方法和支持，以及过程本身的开发、评价和改进。软件工程环境方面，主要还是考虑实现部分环境，完成有限的集成、有限的设施，并力求与别的环境进行通讯，来支持过程指导和过程的半自动化运作，并通过与过程的模拟运作相结合完成过程的改进。

参考资料

[1] C. J. Tully, Prospects for Future Environments: Introd. to Panel Session, In Proc. of the 9th Intl. Conf. on SE, Monterey, California, Apr. 1987

[2] E. Fedchak, An Introduction to Software Engineering Environments, In Proc. of the COMPSAC 1986, Chicago, Illinois, USA, Oct. 1986

[3] M. M. Lehman, Process Models, Process Programs, Programming Suport, In Proc. of the 9th Intl. Conf. on Software Engineering, Monterey, California, Apr. 1987

[4] J. D. Zucker, ALF: Accueil de logiciel futur, In Software Engineering Environments, Volume 3

[5] A. Legait et al., MASP: A Model for Assisted Software Processes

[6] N. H. Madhavji, The Process Cycle, Software Engineering Journal, Sep., 1991

[7] P. H. Feiler et al., Software Process Development and Enactment, Concepts and Definitions, In Proc. of 2nd intl. Conf. on Software Process, 1993

[8] J. Lonchamp, A Structured Conceptual and Terminological Framework for Software Process Engineering, In Proc. of 2nd intl. Conf. on Software Process, 1993

[9] M. H. Penedo, W. Riddle, Process-sensitive SEE Architecture Workshop Summary, Software Engineering Notes vol 18 no 3

[10] 麦中凡、张莉，集成 CASE 的集成模型，软件学报，1994. 2。

[11] 麦中凡、熊璋，兆程序设计与软件过程驱动的软件开发，第三届抗恶劣环境计算机学术会议，1994，5，洛阳。

[12] Y. Sugiyama et al., OPM: An Object Process Modeling Environment, In Proc. of the 5th Intl. Software Process Workshop, 1990

[13] Y. Sugiyama et al., Describing Working Environments in OPM, In Proc. of the 5th Intl. Software Process Workshop, 1990

[14] An Overview of the ALF Project, In Proc. of the 5th Intl. Software Process Workshop, 1990

[15] Peiwei Mi et al., Process Integration in CASE Environments, IEEE Software, 1992, 3

[16] R. N. Taylor, et al., Foundations in the ARCADIA Environment Architecture, Software Engineering Notes, 13(5), Feb. 1989

[17] M. Dowson, ISTAR and the Contractual Approach, In Proc. of 9th ICSE. USA, 1989

（上接第80页）

的语义数据模型，它是一种很有前途的方法。由 Coad 和 Yourdon 方法创建的 OO 规范说明的派生，说明了该方法的进化是有实现可能的。

向面向对象的过渡，主要依赖于系统开发者确信该方法的好处。由于没有一种 OO 方法包含明确的步骤或模型用于重用系统开发，所以对象模型并不能完全解决重用问题。

面向对象的软件体系结构将在九十年代占据主导地位，向这种新型体系结构的过渡正在进行，现在已出现了面向对象的语言、数据库、界面、操作系统及开发环境，而且新的数据类型、分布式处理、多媒体应用程序以及端点用户计算正在有力地推动着面向对象软件的实现。

主要参考文献

[1] 马茜等，面向对象的软件设计基础，北京科海培训中心

[2] Booch, G., Object-Oriented Development, IEEE Trans. Soft. Eng., Vol. 12, No. 2, 1986 pp211—221

[3] Jackson, M., Systen Development, Prentice Hall Int. UK, 1983

[4] Cameron, JR., JSP and JSD, the Jakson Approach to Software Development, IEEE Computer, Society Press, USA, 1988

[5] A. G. Sutcliffe, Object-oriented System Development: Survey of Structured Method, Information and Software Technology, Vol. 33 No. 6, July/Aug. 1991