兆程序设计 程序设计 软件过程 软件开发

计算机科学 1996 Vol. 23№. 1

44-48

兆程序设计与软件过程驱动的软件开发*)

TP311 TP311.52

麦中凡 熊 璋

(北京航空航天大学 北京 100083)

摘 要 Megaprogramming and software process driven developing is a new generation of software engineering technology. This paper introduces the background of the technology, and points out that software development needs to solve two problems: to build an environment for and to choose a method of software process development. Megaprogramming fully embodies the methodology of software process driven, which has the conventionalization as key job, ie., the evolutive process programming. The last part of the paper is focus on the achivement and existing problems of megaprogramming.

关键词 Megaprogramming, Software process, CASE, IPSE, Distribution, Interoperability, Component interconnection.

1. 引 言

基于软件工程过程(Software Engneering Process)的兆程序设计(Megaprogramming)是 90 年美国 TR1-Ada 会议上 DARPA 提出的新的软件工程概念,反映了软件技术的最高成就。大体说来,软件技术发展经历了如下五个阶段:微程序设计、低级程序设计、高级语言程序设计、面向对象程序设计和兆程序设计。兆程序设计用 GUI 构件器、UNAS、电子报表之类用户界面工具,改编各软件成分和硬件设施(Components & Services)组成新的应用系统。它是在软件处理过程可控情况之下,复用已有资源,即时调整、管理、优化软件工程过程(此时应叫兆程序设计过程)而生成的应用系统。

当今分布在异质网络上的多媒体应用系统动辄规模即达数百万行源代码,兆程序设计从方法学、范型、支持环境都和过去有着质的飞跃。然而,对于我国广大软件工作者,兆程序设计却是一个崭新的观念。我们早已认可基于结构开发的瀑布模型(生存周期)的软件工程,也意识到静态阶段开发必然为多次返复原型化开发所替代。面向对象被认为是支持重用、支持原型开发最有生命力的新方向……怎么又出了了一个兆程序设计?它的本质是什么?和当今软件工程热门技术的关系是什么?它的生命力如何?有没有成熟的应用范例? 本文试图解释这些问题。

2. 技术背景

对软件本质的研究表明,软件的可修改、易维护、可移植、可重用等一系列影响结果软件的性质寓于开发方法、表示、开发过程之中。结构化、数据抽象和信息隐藏、面向对象以及与之相应的程序设计语言(Pascal、Ada、C++)使软件结构、表示、开发方法日趋完善,并发展了相应的方法学。兆程序设计就是在软件工程环境相对成熟的情况下对软件开发过程深入研究之后提出来的。以下分述软件工程环境的发展和软件开发方法学的进展。

2.1 软件工程环境

软件工程环境(SEE)是一个概念上的术语,是计算机辅助软件工程环境(CASEE)和集成的程序设计支持环境(IPSE)的泛称。CASE 环境是软件厂商为软件工程的某个阶段,在自己选定的硬件平台和操作系统上,开发的辅助或自动工具。环境即工具

*)本文为自然科学基金项目 6907334"大型软件面向对象分析"的相关论文

[5]扬晓帆等,一类容错多总线结构.计算机科学,本期

[6]C. Berge, Graphs and Hypergraphs, North-Holland Publishing company, 1973

集，由市场的需求推动，虽价廉，适用，但不通用。随着工具的发展，出现了跨生存期两个或多个阶段的工具，不同机型相互转换的工具，仍至80年代后期提出的集成的CASE(I-CASE)环境。

IPSE源自1984年Ada程序设计环境APSE的石人计划的思想[1]：为程序设计提供全生存期的支持。强调集成性、可移植性和开放性。着重研究环境的低层结构，即定义集成的框架(framework)。解决数据集成(所有工具、软件产物、数据均用统一的观点存取与维护)、表示集成(在单一用户界面上可以无歧意地操纵环境中的任何成分)。控制集成(按某种框架的模式、控制工具、软件产物的运行)。IPSE往往是举国大量投资的基础项目，如美国的APSE及以后的STARS计划；英国的Alvey计划中的Aspect、ECLIPSE和IPSE2.5；欧洲共同体ESPRIT计划的尤里卡软件工厂(ESF)和ARISE[1]等。其费用以千万甚至数亿美元计。

IPSE不同于CASEE，一开始就从整体上研究集成模式，即至少有三个层次的软件设施：在各个不同硬件平台和操作系统上先作一可移植层；然后是集成框架层；最后是各种工具的一层。美国和欧共体在第一层上就花了3—4年的功夫。在APSE的核环境上做通用Ada接口集(CAIS)，旨在所有用Ada编写的程序和工具不加更改地在不同的机器上运行。欧共体则开发可移植通用工具环境(PCTE)，这种通用的接口远不止传统数据格式转换意义上的接口，而是任何机器按接口标准配备了接口层就提供某种(如Ada或C)语言的虚机。用统一的表示即可操纵所有软、硬件资源。软件独立于硬件，可先于(照标准即可)硬件开发。这是一项难度较大且意义深远的工作。1986年CAIS和1988年PCTE成为标准DOD/STD-1838和PCTE149[2]。

整个80年代，蓬勃发展的软硬件新兴技术对集成的软件工程环境有深刻的影响。主要是网络、面向对象、多媒体、原型开发方法学。硬件元件的廉价和网络系统高的性能价格比，使得单主机上的IPSE的移植问题变为在异质机网上可互操作问题。面向对象的封装性、继承性和通信机制天然支持网络系统上的数据集成("数据"封装为对象并相互通信，内部实现可在不同结点上以不同语言、格式表示，并由各结点自己解释，实现虚机层上的数据集成)。面向对象支持多媒体的二进制大对象块BLOB，天然地支持界面上的表示集成。面向对象对实现控制过程框架也带来了极大的方便性(工具或其它软件实体通过对象要求代理人(ORB)以对象身份和其它"对象"通信)。因此，面向对象技术很快地渗透到集成的软件工程环境，使得基于实体-关系数据集成的CAIS标准显得落后，PCTE转向以对象描述的PCTE(一)。不仅各IPSE，各厂商或其联合追求的集成CASE(即I-CASE)，在其环境系统中都增加一对象管理层(OMS)。

除此而外，封装的对象还支持开放式环境，即每个工具经封装后作为"软插件"加入并扩充该系统。开放式系统随着时间推移日益完善，但对集成框架的管理要求更高。于是，欧洲制造商协会(ECMA)于1988年提出集成框架的烤面包模型[2]。

这种框架模型的核心是信息仓库，各厂商也按类似的框架研制自己的基于仓库的I-CASE。典型的有IBM的AD/cycle，HP的SoftBench，DEC的COHESION-CDD/Repository。微机市场上Microsoft的产品如Windows-NT，Cairo也向它靠近(只是表示和控制集成，数据集成几乎没有)。即使是大公司的系统，由于信息模型(即开发过程模型)比较死板，所以只支持某一既定领域(如数据处理)应用。好在这个领域软件需求量大，这种环境还是成功的。

有了这种环境，80年代后期在数据处理领域提出了"信息工程"的概念[3]，大量利用可重用成分(80%以上)和自动代码生成工具完成应用信息系统。软件生产率成10倍地增长。

但是对规模庞大、处理复杂又经常修改的大型军用系统，这种IPSE投资既大，修改又不灵活，人们常常在不太合适的软件开发过程模型下开发出不尽人意的产品。IPSE备受责难，导致了下文还要详细讨论的软件过程模型的研究。

2.2 软件开发范型的发展

传统的基于瀑布模型的软件开发范型，其阶段不可逾越，发现错误晚，交工晚导致费用损失巨大，自然就转向生存周期各阶段可多次返复的原型(prototyping)，旨在开发出能反映目标产品主要特征的原型，借助可重用件和辅助工具，快速开发出可演示的原型系统，然后扩展为产品[4]。在快速发展CASE工具的80年代，原型几乎成为软件开发的主要范型。但在大型系统中原型系统暴露其弱点，主要是：(1)模型效应，对开发者不熟悉的领域易于开发出不切题的软件框架。(2)原型迭代不收敛于预先制定的目标。(3)资源规划管理较难，随时更新文档带来工作量大。(4)依赖开发环境，不易适用于嵌入式软件、实时控制软件、科技数值计算。

IPSE 基于某个过程模型集成，对符合该过程模型的开发，支持是有效的。然而，过程模型有了变动（见下文）支持就不太有效了。原型虽则提供构成合理软件框架的可能，但能不能使这种可能更加有效呢？这要对软件开发过程模型进一步研究，提供一个"一定会"开发成良好体系结构的过程（演进）模型。在美国，80 年代中期软件开发过程模型研究和大型程序设计开发方法研究同时并举。1988 年 B. Boehm 等人提出螺旋（Spiral）开发模型，其示意如图 1。

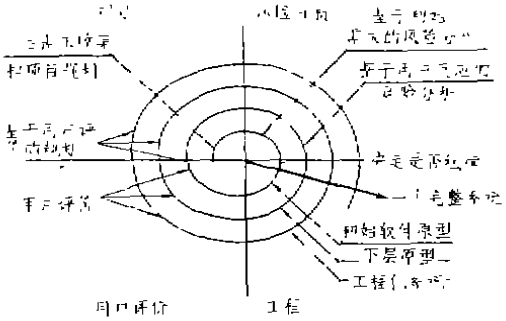


图 1　软件开发的螺旋模型

螺旋模型把软件开发过程看作是如同阿基米德螺线的逐步演进（扩大）的阶段作业的反复。规划—风险分析—原型（或产品）开发—用户评价四个阶段周而复始，即每评价一次原型重作一次规划和风险分析，使"原型"能收敛于既定目标，且软件工程项目管理得以实施。经费，进度综合调整。如果风险分析结果令人失望，则夭折此项目，否则作最后工程性开发直至交付。螺旋模型正好克服了原型在大型项目中的缺点，但同时应该看到这是增加了多次人工干预（重修计划、评审）的结果。

3. 软件过程

3.1　软件过程及其成熟度

所谓软件过程即开发系统（包括与开发活动有关的一切人、物）的一系列活动，瀑布模型给我们以清晰的软件开发过程（Processes）的概念，即一个项目要经历需求分析、初步设计、详细设计、编码测试、交付验收等一系列软件工程活动的过程才转入使用维护过程。但过程远不止于此。例如，上述螺线型开发就是一个上层的过程模型，其过程周而复始地进行。此外，应用领域不同，重用件大小、多少，实现的硬件基础（分布式异质网、单主机）和软件规模大小都会影响到一个项目的开发全过程。过程模型是抽象的，具有一定的共性，而每个项目的软件过程是具体、可描述、有差别的。过程好坏对于软件开发的质量和成本至关重要。

最先认识到这个问题的是美国国防部软件工程研究所（DOD SEI）和卡内基·梅隆大学的一批研究者。早在 1985 年，R. A. Radice 就发表了"程序设计过程研究"、"程序设计过程体系结构[5,6]研究报告，不过，这些成果最初只用来代表美国国防部调查各企业软件过程的成熟度。以 W. S. Hamphrey 的"刻画软件过程：一种成熟度的框架"一文最为典型。以后并以它作为评估一个单位（或公司）软件过程能力的标准，它把成熟度分为初步过程、可重复过程、可定义过程、可管理过程、可优化过程五级，而且根据各级刻画的指标证实各级间不可逾越，必须在人员水平、管理水平和软件支持环境上逐级完善提高。不幸的是，调查美国国防部的承包商，大多处在第一级（86%），少量第二级（12%），只有 2%的承包商达到第三级。到 87 年，第四、五级企业还没有[7]。从以下指标可以看出各级应具备的条件（上一级包括以下所有各级指标）：

（1，2）可重复级以下：建立了过程小组；建立了软件开发的体系结构；实施软件工程方法和技术；有项目管理和质量保证。

（3）可定义级：建立了有质量和费用参数表征的过程管理的基本集合；有过程数据库；能采集和维护过程的数据；能评估每个产品的有关质量并通知管理。

（4）可管理级：支持过程数据的自动收集；能利用收集到的数据分析和修改过程。

（5）可优化级：能连续改善和优化过程。

然而，一个单位（或公司）要作以过程为中心（或称过程驱动、基于过程模型、面向过程）的软件开发，必须要能定义、管理、控制（或称优化）过程，即达到第三、四、五级。

3.2　以过程为中心的软件开发

以过程为中心的软件开发的基本思想是结合描述开发产品的体系结构，逐步精化，开发出最优化的过程。然后由过程环境利用重用构件自动生成项目产品的各成分，其步骤是[8]：（1）了解问题当前软件过程的情况，作过程分析。（2）定义和开发本项目要求的过程（用适当的图形或过程描述语言表示）。把过程定义为任务序列。要考虑各任务间关系，所用工具和方法、技艺、人员调配、培训。（3）建立完善各过

程的活动表，并按优先级排序。(4)作出完成这些活动的计划。(5)提交实施计划的软、硬件资源，由机器自动(程序生成器，重用构件块)或人工完成项目软件的开发。简言之，要解决两个问题：一为支持过程开发的环境，一为过程程序设计方法学。

80年代中期美国国防部尖端防御研究项目署(DARPA)支持了一批这方面的研究。如研究APSE的CAIS的一批人继续研究Arcadia过程为中心的环境。集合了六所大学和公司平行研究五个子项目：环境体系结构；软件过程；软件对象管理；用户界面；度量和评价。1990年发表的"Arcadia-1过程中心软件环境的体系结构"[9]一文是这方面的重要成果。该文透露的Arcadia-1虚机图是环境的总模型，利用面向对象的客户/服务器技术把整个环境看作是分布式硬件平台上的四层虚机。环境构造者用虚操作系统界面建立四个服务器：用户界面管理(UIMS)系统、对象(OM)管理系统、测量与评价和过程实现机制。在环境层界面上提供对应的接口。过程程序员以这四个接口提供的资源建立实现UIMS客户界面，OM界面、过程程序设计语言、测量与评价描述界面，过程描述界面等上层接口的服务器；UIMS客户构造器，对象定义工具，测量与评价服务器工具，过程程序设计解释器，过程实现。项目(产品)程序员在最上层虚机上利用五个接口提供的资源工作，开发应用。类似的系统还有TRW的泛网络体系结构设施(UNAS)。

在基于过程开发的方法学方面，DARPA支持的STARS计划也有许多项目。与此同时，民间对软件过程研究也异常活跃，国际软件过程专题会(ISPW)自84年以来已开过七次。过程驱动、过程中心开发对"如何定义过程"始终是热门题目。普遍认为过程定义必须形式化，因为这样便于与外部用户、内部同仁讨论、修改，也宜于自动生成目标项目。至第六次ISPW会议(1991年)已提出形式化方案18种[10]。集中于三种途径：抽象层次图、过程程序设计语言、Petri网。结构图的好处是直观沿袭传统，但语义缺乏通用性，动态性，如美国空军开发的IDEFO就从SADT表示法演化而来。过程程序设计语言大多定义为Ada语言的超集，如马里兰大学的MVP-A，科洛拉多大学与STARS合作的Ada过程程序设计语言(APPL/A)、它的好处是直接运行，易于扩充，是目前使用较广的方法。Petri网方法虽然在一定程度上能表达软件过程活动的动态性和分布性，但数学太深，一般过程程序员不易掌握。比较有希望的研究方向是系统动态的方案，既有Petri网的表达能力又不那么数学化，但目前还没有。

4. 兆程序设计

兆程序设计的本质是从预先集成好的、可靠的、可重用的、异构的、分布在异构网上的软件成份构造出大型软件系统，其构造方式是演进式的，构造出的产品是可再次重用的产品系列。

所谓产品系列是指如同类属程序包那样的产品，输入不同参数可得到一系列程序包。正是由于面向域的开发，它需要有特定域的软件体系结构(DSSA)。借助DSSA开发，开发产物又丰富了这个DSSA。这个DSSA中的各成分相互连接就构成了它的框架。从设计的角度，兆程序设计从已有DSSA框架"摘取"、"补充"使之成为满足需求解的框架。所谓互连框架(也称模块互连框架MIF)，它包括(1)体系结构描述；(2)系统配置描述；(3)在异质环境上运行时刻对可互操作性的支持描述。

兆程序设计的中心工作是为框架定型(conventionalization)即演进式过程程序设计的完成。已有(可重用体系结构的)框架被认为是定型的。设有一应用软件要开发，其步骤是：

(1)首先作软件问题的域分析；(2)定义问题区间需求；(3)作共性和差异分析(哪些成分重用、修改、新设计)；(4)为满足需求的解的框架定型，定义实现解的体系结构的过程(演进式的)；(5)实现解的体系结构(在此期间按过程完成风险评估、质量度量、新构件生成、构件测试、总体测试。请注意，它们也是演进式的)。

之所以求解一个问题区间，是为了得到产品系列的结果，否则产品不能再次重用，这也是重用的新观念[8]。

由于一个问题区间的解，其中重用成分的多少、层次深浅是不同的，而每个重新开发的新成分其软件过程也不尽相同。为求得最小风险最好结果，兆程序设计天生就要依靠最佳软件工程过程。因而，必然是以过程为中心。开发子过程和管理子过程(某个阶段活动)混合都利用工具(软件)实施(执行)。这些不同工具和技术支持(如数据库)原本分散在异质网的结点上，过程程序设计为它们提供组合、调度、实施的兰图。因而，能即时报告开发情况，求得最小风险。

显而易见，兆程序设计要求良好的可视性。用户代表，经理，不同层次开发人员要随时在自己终端前

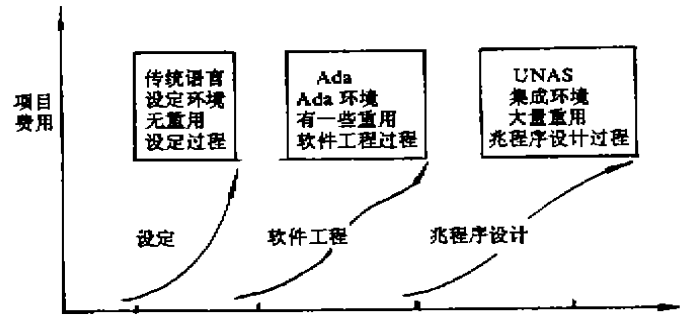
看到开发情况。限于各软件资源，当今还不能做到统一程序设计语言，其环境应支持多种语言，并向多媒体表示演化。

5. 兆程序设计的成就和存在问题

兆程序设计是目前蓬勃发展的大型软件技术，和其它计算机技术一样：理论不完备，原型在开发，商品已面市，它能为应用该技术的人带来利益，在DARPA的许多项目已进入实用，例如，大型实用的例子有：1989年欧洲23个国家民航部长会议决定成立欧洲空中安全导航组织(EUROCONTROL)，并拟定中心流管理单位(CFMU)计划。其战术子系统(TACT)约16万行Ada源代码拟于94年完成。它用美国TRW的泛网络体系结构设施(UNAS)产品描述其原型，采用兆程序设计[11]。又如，美国指挥中心处理和显示系统(CCPDS-R)由美国空军系统命令指挥部电子系统分部和TWR联合开发，分三个子系统，第一个为公用子系统，约30万行Ada源代码，27个月完成(至93年)，其余在开发中。它是高可靠性，实时分布式系统，具有综合的用户界面，以TRW的Ada过程模型作增量式开发[12]。

也许有人以为如此Ada大项目我国短期内不会投资开发，过程中心的兆程序设计离我们还很远。事实上，美国苹果公司的AppleEvents和Microsoft的DDE等对象管理组(OMG)的系统已采用兆程序设计。

下图显示了兆程序设计的规模经济性：



体系结构:	单程序 单平台	多程序 单平台	多程序 分布式平台	大型，异构 分布式系统
设计范型:	功能设计 瀑布模型	基于对象 2167A	面向对象 演进式	兆程序设计 演进式
平台技术:	独占式	处理器族	形式系统应答	形式系统应答
规模:	小	中等	大	极大

图2 不同工程方法的项目经济性

以下是传统开发方法和兆程序设计对软件品质影响的对比表：

影响质量	传统的	兆程序设计
需求未知处	发现晚	解决早
开发风险	很晚才知道	解决早
可重用性	不支持	本性
更改管理	混乱和"维缠	直接和顺当
设计错误	发现晚	解决早
授权性能	前所未有	实际可行
费用预计	不可测	可决断
时间表	不可缩减时常拖期	质量、性能、功能、技术均可调整
目标资源估计	由定量模拟，分析	由执行原型定量

兆程序设计的意义不仅为人们对日益增大的系统的开发方法指出出路，更重要的是它和传统软件工程一样展示了新一代软件技术。它的不足正是向计算机科学家提出的课题[13]：

(1)如何构成解的体系结构？现在是凭人的经验，根据已有DSSA构造，是否真切题？能不能找到一般性？过程程序设计相当于常规编程结构化以前的时代。

(2)体系结构理论，成分互连形式化。

(3)表达体系结构和过程综合的计算语言，要能执行，支持原型过程。

(4)应用域要更多，才能找出一般性。至少找出某一应用域内软件体系结构一般性。

(5)过程中心环境还不够成熟完善，包括封装、容错、通信、派定调度、实时、效率。

(6)投资大，人员培训、技术转轨有一定难度。

(参考文献共13篇略)