·方法论·

从结构化方法学走向面向对象方法学

赵玉鹏[1]　梁国钊[2]

(1. 广西百色市右江民族医学院,广西 百色 533000;2. 广西大学社会科学与管理学院,广西 南宁 530004)

摘　要: 计算机软件设计中最主要的方法是结构化方法和面向对象方法,它们在许多方面都有明显的区别,其发展过程反映了软件开发从以机器为中心到以人的认识为中心的转向、一场信息产业界的技术革命和软件设计思想的改变。

关键词: 软件开发方法学;结构化方法学; 面向对象方法学

中图分类号:B026　　**文献标识码**:A　　**文章编号**:1003 - 5680(2004)05 - 0048 - 04

引言

结构化方法和面向对象方法是计算机软件设计中最常用的两种方法。结构化方法在一定程度上解决了软件的可靠性、可维护性和可理解性等问题,而 20 世纪 80 年代中期以后,面向对象方法开始风靡全球,被广泛应用于计算机软件的各个领域,如计算机仿真、系统设计、图形处理和人工智能程序设计等各个方面,进而深入到计算机硬件设计及其它工程设计领域,显示出其强大的生命力。从结构化方法到面向对象方法,表面上看是软件开发工具在城头变换大王旗,实质上反映了一种深刻的认识论的变革和一种技术发展的模式的演化。

一　软件开发方法学诞生的根源

软件开发方法学主要是由于人类认知的有限性与客观世界的无限复杂性以及计算机应用的广泛性而产生的。

1. 认知可计算理论表明人的认识能力是有限的

人们总是希望确定的知识,渴望超越千年而至永恒的知识。维特根斯坦是 20 世纪乃至人类哲学史上最伟大的哲学家之一,他认为语言不能把握世界上存在的全部东西。作为维也纳文化上最富有创造力的那个时代的学生,哥德尔深受语言哲学的影响,于 1931 年无可辩驳地证明,存在着可以被看作是真却不可能被证明是真的数学命题。哥德尔不完全定理永远地击碎了希尔伯特所抱有的完全彻底地给出算术的公理化 ——从而数学的公理化的一线希望,对数学、哲学、计算机科学、语言学和心理学等产生了巨大的影响。

哥德尔等人提出了原始递归函数,后来又提出了一般递归函数的概念。丘奇 —图灵论点是递归论中最重要的结论,递归论又称可计算理论。丘奇论点这样叙述:每个能行可计算的函数都是一般递归函数;图灵论点这样认为:能行可计算函数都是用图灵机可计算的函数,与一般递归函数是等价的。

丘奇 —图灵论点的提出在数学和计算机科学上有着重要的理论意义和现实意义,对于计算机科学,它明确地刻画了计算机的计算能力,确定了计算机只能计算一般递函数。除了理论可计算外,还存在着一个现实可计算问题。

丘奇 —图灵论点的哲学意义在于它表明了人类认知计算主义本质。丘奇 —图灵论点从人的认知结构、认知过程和认知能力三个方面综合刻画了人的认知本质,表明了人的认知结构是一种递归结构,人和认知过程是一种递归计算过程,人的认知能力受递归规律的制约。

2. 客观世界是复杂的

在解决了什么问题是可计算的,什么问题是不可计算的之后,接着人们要解决一个计算复杂性的问题。对可计算的和不可计算的函数进行分类:从最简单的到较复杂的,从较复杂的到最复杂的。

分层理论的核心问题是利用递归论工具给出数集(问题集)或函数集的复杂性的某种排序。分层论表明:无论是谁,也不管将来认知条件多么好,甚至无论人智力进化到哪种程

【收稿日期】 2003 - 09 - 30

【作者简介】 赵玉鹏(1974 -),男,河南鹤壁人,广西右江民族医学院社科部讲师,科学技术哲学硕士,研究方向为科学方法论;
梁国钊(1940 -),男,广西容县人,教授,硕士研究生导师,研究方向是科学方法论。

度,来自客体自身的客观性对认知程度的影响也不会消除。推而广之,人们对问题的解决或认知程度也是分层的,即有的问题可以部分解决,有的可以部分解决或认知,而有的人则完全无能为力。

我们生活在一个复杂的非线性世界里,处在有序和混沌的边缘。复杂性和非线性是物质、生命和人类社会生活的显著特点。

3. 计算机应用的范围是广泛的

电子计算机在 20 世纪中叶诞生后,迅速从科学与工程计算应用到工业自动化、商业和管理等各个领域。相应地,计算机软件开发也越来越庞杂。

每一种软件设计方法莫不都是希望在开发大型复杂软件时,能够掌握隐藏在其背后的高复杂度,为此计算机界许多人致力软件开发方法学的研究。

二 结构化方法学与面向对象方法学的不同源头

结构化方法学与面向对象方法学都是在 20 世纪 60 年代产生的,但是原因却是不同的。

1. 软件危机导致结构化方法学的诞生

20 世纪 60 年代初以前,程序编写的问题似乎掌握在程序设计师手中,到了 60 年代中期以后,随着软件规模的扩大和复杂性的增加,软件的开销(开发时所花费的人力、物力和动行时所占用的硬件资源及运行时间)也惊人地增加了,软件的可靠性、可维护性与可理解性却明显地降低了,人们惊呼这是一种危机!最明显的一个例子是 IBM 开发 OS/360 操作系统的失败。对于一般大众而言,由于软件错误而导致的灾难当中,印象最深刻的莫过于"水手一号"事件。

于是,为了设计出更好的软件,人们开始寻找更好的开发方法,其中结构化方法学得到最多人的采用。结构化方法学认为,驾驭复杂的最佳方法,就是将软件设计人员的视野限制在较小的范围之内。因此,软件的开发程序应该是:程序设计师先行浏览软件作品的轮廓,自觉能够掌握之后,便将其置于一旁。此时,所有程序设计师都将注意力集中于比较低级的细节上。如此反复进行,直到程序设计过程中,所有细节都能找到合适的语句编写为止。结构化方法是一种相当良好的直觉工程训练,同时也是 20 世纪 70 年代最成功的一种软件设计方法,这时候的程序设计语言如 FORTRAN,COBOL,PASCAL 等都是结构化程序设计语言。

2. 面向对象方法学首先在运筹学领域产生

20 世纪 40 年代初期,由于军事上的需要产生了运筹学。二战以后,运筹学的研究和应用由军方扩展到民间,这门学科的理论体系也不断完善。50 年代以后,随着电子计算机技术的迅速发展,运筹学的很多方法如线性规划、动态规划和网络模型等由于能够更加方便地求解被进一步应用于实际管理系统的优化问题。

20 世纪 60 年代,挪威计算机科学家奈加特大学毕业后分配到挪威国防部,从事程序设计和运筹学方面的工作,后转至挪威计算中心。运筹学研究中的首要问题是为系统建立数学模型,而模型所要解决的首要问题是如何描述系统中的不同组成部分及其运行。20 世纪 50 年代时通常通过符号标志(Symbol Notion)实现,例如用流程图加上若干系统运行规则,然后用蒙特卡洛法加以分析,使模型逐步完善,这种方法很不方便,效率不高,他要寻求一种新的有效的方法。到 1961 年前后,形成了一些清晰的概念,但凭着他的计算机知识和经验不足以设计出新的计算机语言去实现他的设想,这时他的老朋友戴尔也来到挪威计算中心,戴尔在计算机程序语言方面有相当丰富的经验和深厚的功底。这对"最佳搭档"于 1962 年推出了 Simula 语言。

由于 Simula 语言是为解决运筹学也即在系统工程存在的问题而设计的,所以特别适用于研究售票系统、生产线组织、神经网络、并行程序处理等离散事件。

他们对取得的成绩并不满足,他们的目标由专用语言转向了通用语言、对象、类和子类等基本概念也正在这个时候出现和形成,从而诞生了第一个面向对象的程序设计语言 Simula67。20 世纪 80 年代以后,又相继出现了 Smalltalk-80 和 C++等面向对象的程序设计语言。

三 结构化方法学与面向对象方法学的比较

结构化方法学和面向对象方法学异曲同工,都很好地促进了软件工程的发展,但他们在许多方面都是不同的。

1. 哲学思想的比较

结构化方法学认为世间万物不论是原子、电子、山川、日月星辰和动植物等都是由变动不居的事件或过程所构成。这些事件或过程相互依存、相互转化而形成统一世界这个有机体总的过程体。

而面向对象方法学认为世界是由不同层次的对象组成的,每种对象都有各自的内部状态和运动规律,不同对象的相互作用构成了不同的系统。

2. 技术思想的比较

结构化方法是将一个复杂的大型系统分解成一个个简单的系统,用系统工程的思想和工程化的特点,按照用户至上的原则,结构化、模块化和自顶向下地分析和设计。顺序、分支和选择三种结构都可以构成任何计算机可以处理的模块。

面向对象方法吸收了结构化方法的主要优点。它将数据与操作放在一起,作为一个相互依存、不可分割的整体——对象来处理。它将对象及对象的操作抽象成为一种新的数据类型——类,它综合了功能抽象和数据抽象,使用了信息隐藏技术——封装。

3. 基本概念的区别

结构化方法学里最常用的概念是系统流程图,数据流程图,模块等。

系统流程图是描绘物理系统的传统工具,表达的是信息在系统各部件(程序、文件、数据库、表格和人工过程等)之间流动的情况。

数据流图是描绘逻辑系统的工具,图中没有任何物理的元素,只是描绘信息在系统中流动和处理的情况。

模块是组成结构化软件的基本单位,能够完成一项独立

的功能。

面向对象方法学里最常用的概念是对象、类、封装、继承、消息、方法。

对象是具有特殊属性(数据)和行为方式(方法)的集合体。数据是对象的静态特征,而方法是对象的动态特性。

类是具有相同或类似属性和方法的对象的集合体。一个类的上层可以有超类,下层可以有子类,形成一种层次结构。

继承性是自动地共享类、子类和对象中的方法和数据的机制。

封装机制,它是一种信息隐藏技术,用户只能看到对象封装界面上的信息,对象内部对用户是隐藏的。封装的目的在于将对象的使用者和对象的设计者分开,使用者不必知道行为实现的细节,只需使用设计者提供的消息来访问对象。消息是对象与对象之间进行通讯的工具,发送消息的对象称为发送者,接收消息的对象称为接收者。消息告诉对象要求做什么,而不管采用何种方式做。

方法是对象的动态特征,它能够响应对象发送过来的消息,采用何种方式实现其功能,以及如何改变对象的属性。

4. 程序设计方法的区别

结构化程序设计方法着重于过程,"程序设计 = 算法 + 数据结构"是其基本指导思想。在结构化程序设计中其有严格的理论基础,结构化方法学的创始人 Dijkstra 在 1972 年讨论了三种支持结构化程序设计的数学推理方法:枚举,推理和抽象。

而在面向对象程序设计中以对象为中心,"程序设计 = 对象 + 类 + 继承"是面向对象程序设计的基本原则。面向对象方法的基本概念如类、对象、继承等虽说在一定程度上有了形式化描述,但总的讲来其缺乏一种统一严格的数学理论基础。

5. 软件开发过程的区别

软件开发的过程就是利用计算机语言将人们关心的现实世界的问题映射到计算机世界的过程。可以这样描述:现实世界 ——建立模型 ——编程实现 ——计算机世界执行求解。

在结构化设计中,程序员分析了问题域之后,得到了一个面向过程的模型,其实现过程可描述为:

现实世界 ——流程图 ——面向过程语言 ——执行求解。

结构化方法有三大缺陷:一是没有一个联系各个阶段的统一模型;二是后期的变化和改动困难。三是不支技术重用。

在面向对象方法设计中,其实现过程为:现实世界 ——类图 ——面向对象语言 ——执行求解。

面向对象方法软件设计中,对象的概念弥漫着整个从分析、设计到编码的开发过程。对象和它们之间的关系成为各个阶段的共同表达媒介。开发的重心由编码向分析偏移,从以功能为中心转向以数据为中心。开发过程的迭代和无缝性使得重用变得更加自然。

四　从结构化方法学走向面向对象方法学的原因和意义

结构化方法学在 20 世纪 70 年代计算机软件设计中曾经风靡一时,到了 20 世纪 80 年代中期以后,其霸主地位逐渐被面向对象方法所取代,有其重要的原因和深刻的意义。

1. 从结构化方法学走向面向对象方法学的原因

首先,从结构化方法学走向面向对象方法学在技术上有其可行性。

虽然结构化方法学和面向对象的思想都是起源于 20 世纪 60 年代,但是面向对象方法并没有很快流行。一方面是因为面向对象方法学是随着面向对象的语言而发展起来的,第一种面向对象的语言 SIMULA67 固然难学难用,而更重要的是,面向对象的语言由于增加了类、对象、封装和继承等概念,需要较复杂的硬件的支持。

由于类的继承作用封装机制,人工编写的代码越来越短,而可执行文件变得越来越长。

即使如此,人们还是发现运行速度快多了。究其原因,是由于在短短的几十年时,计算机业界发生了巨大的技术革命。电子计算机从诞生到现在,经历了从电子管到晶体管,从晶体管到集成电路以至于超大规模集成电路阶段,从 20 世纪 60 年代开始,摩尔定律就一直在微处理器设计中起作用,其表明,每隔 18 个月,计算机的运行速度就上翻一倍;另一方面,其成本却要下降一倍。

这种革命不仅反映在硬件技术上,而且还表现在软件技术上,各种基于面向对象的自动生成软件开发工具在 20 世纪 90 年代相继浮出水面。

现代计算机是超大规模集成电路、人工智能、软件工程和新型计算机体系结构等综合而成的产物。由于硬件和软件两方面的共同努力使得从结构化方法走向面向对象方法从技术上可行。

其次,面向对象方法能够构造出更好的软件系统。

结构化方法在本质上具有冯·诺依曼机系统的结构特点:面向过程,即以"过程"和"操作"为中心来构造系统、设计程序,这样思维成果的可重用性必然较差,究其原因,在于人必须先知道一个事物是什么,然后才能觉察这个事物中所发生的变化,即过程。经验也告诉我们,在客观世界以及作为它的映射的软件系统中,"过程"和"操作"是不稳定的,多变的,而"对象"和"数据结构"则相对稳定得多。换言之,若以"对象"和"数据结构"为中心来构造系统、设计程序,则思维成果的可得用性就有可能较好。

2. 从结构化方法学到面向对象方法学的转变反映了从以机器为中心到以事务为中心的转向

1936 年,计算机的开山鼻祖英国数学家图灵发表了一篇论文《论可计算数及其判定问题中的应用》,从而奠定了计算机的数学理论基础,十年后计算机问世,现在的计算机体系结构成为其实际计算模型。传统的计算机虽然有各种各样的体系结构,但都未能从根本上改变冯. 诺依曼机的一些基本特征,即顺序地指行指令,按地址访问线性的存储空间,数

据和指令在机器内部采用统一的表示形式,以及数字计算。

结构化方法学更接近计算机的物理世界。结构化方法强调按照计算机本身的特点去设计软件。为了编写一个程序,我们必须考虑内存、堆栈、指针、I/O 地址等,我们不得不知道电脑实现这个程序的过程,然后用一条条的指令去完成。举个例子来说,做一个中等规模软件的界面设计,我们如自然语言几句或几十句话即可清楚地描述出来,而编写的程序却需要耗费大量的精力,仅程序代码就有数千及至上万行。

20 世纪 80 年代以后,面向对象方法流行以后,这种情况有了很大的改观。同样是设计做一个界面设计,采用面向对象的方法使用"类库"实现,则只需要片刻功夫即可解决。而这时开发人员可以把大量的时间应用于具体的应用领域的分析和设计工作。

3.从结构化方法学到面向对象方法学的转变消除了形式化与非形式化之间的鸿沟,体现了形式化与非形式化之间的辩证法

多年以来,在计算机软件界存在这样一个争论:是走严格的形式化(数学公理化)道路,还是走松散的非形式化或半形式道路?形式化道路要求软件从业人员有较深的数理功底,编写出来的程序代码短小精炼。而且半形式化和非形式化则是面向大众化的,进行软件设计则主要靠的是经验,更侧重于实际问题的研究,程序设计者往往只知其然而不知其所以然。结构化方法学更注重形式化,而面向对象方法学走的却是一条半形式化以至非形式化的道路。所以有人说,培养一个结构化程序设计员远比培养一个面向对象程序设计员困难得多。可以这样说,由于面向对象技术的日趋成熟,大大降低了软件开发的难度,使得软件开发的门槛大大降低了。

形式化方法能够最大程度地消除软件设计中的错误,这是计算机专业人员所追求的;而非形式化方法却能够更好地容纳各类人员加入软件设计的战团,推动软件事业的发展,这也是我们所希望的。面向对象方法学消除了形式化方法与非形式化方法之间的鸿沟,体现了形式化与非形式化之间的辩证统一。这是因为:一方面任何一种程序设计语言本身就是一种形式语言,面向对象的程序设计语言本身如 Smalltalk-80,C++,java 等也不例外,这种程序语言本身的设计要求非常严格地遵循形式化方法,需要很深的专业功底和很好的数理基础;另一方面这些语言的使用者可以采用形式化方法,同时也可以采用半形式化或非形式化方法。由于这类程序语言的设计者更好地方便使用者,使应用软件开发者像堆积木一样编写程序,所以造就了一大批非专业化程序员。

面向对象方法学给了人们一种自由发展的空间,把从事计算机软件开发者分为三类:一类是计算机专家,这类人员具有深厚的理论功底,在某一方面具有开创性的见解。第二类是计算机专业人员,这类人员具有良好的专业素质,能够根据某些理论编写专业化的软件开发工具。第三类人员是计算机应用人员,这类主要应用现成的软件开发工具从事具体的应用系统开发工作。前两类人员主要运用形式化方法,而第三类人员以半形式化或非形式化方法为主。由于面向对象方法学消除了形式化方法与非形式化方法之间的鸿沟,使三类人员可以很好地合作,有力地促进了软件业的繁荣。

五 结论

结构化方法学伴随软件危机产生,而面向对象方法学首先在运筹学领域里出现。结构化方法学与面向对象方法学各有千秋,它们都是计算机软件发展史上特定阶段的产物,在不同程度上促进了软件工程的发展。我们通过以上分析可以发现软件开发发展到一定程度必然要产生一些科学的方法,各种方法都是齐头并进的,但到底采用哪一种方法,哪一种方法更能适应潮流,是由技术水平、人的思想文化观念和社会等多种因素所决定的。结构化方法与面向对象方法的应用现在均已超出了计算机的软硬件领域,成为一种广泛的通用的方法,如结构化布线系统等等。

随着时代的发展,将会涌现出更多的软件开发方法。法无定法,也许创造方法的人是最愚蠢的。一个网页中这样写道:"自以为是的立法者(人类)犯了一个大错,以为自己能定义整个软件开发过程。他们既不了解其开始,也不了解其结果。学术界试了几下,然后就知难而退了。商业界却对其毫无办法。大型软件开发商们用巨资来制造代码,明明想要扮演上帝的角色,却还装作自己是人类的公仆。所有人都在盯着镜子里的自己,计算机却在一旁偷偷的笑。"

从更深层次的意义上来讲,技术方法都是在实践中产生的。每种自成体系的方法都要比较适应当时的社会生产的需要,到了一定程度就会扩散到其它工程技术领域。

【参 考 文 献】

[1]王荣江. 算法、图灵机、哥德尔不完备定理与知识的不确定性[J]. 自然辩证法研究. 2002(2):48～51.

[2]郝宁湘. 可计算性与不可解性及其哲学底蕴[J]. 自然辩证法研究. 1996(8):23～37.

[3]汪成为,郑小军,彭木昌. 面向对象的分析、设计及应用[M]. 北京:国防工业出版社,1999.10.

[4]约翰. 卡斯蒂等. 逻辑人生——歌德尔传[M]. 刘晓力等译. 上海:上海科技教育出版社,2002.11.

(责任编辑 殷 杰)