XYZ 系统的设计思想*

唐稚松

(中国科学院软件研究所)

THE DESIGN PHILOSOPHY OF XYZ SYSTEM

Tang Zhisong

(*Institute of Software, Academia Sinica*)

ABSTRACT

XYZ System is a CASE environment based on temporal logic and conforming to various ways of programming such as programming with HLL, hierarchical specification or production rules, sequential or concurrent, textual or graphical, etc. All these programming paradigms can be unified with a uniform framework of a temporal logic language. The system is designed with the idea to keep good balance between the theoretical rationalism and the engineering pragmatism. In this paper, this idea is illustrated from philosophical point of view.

§1. 引 言

本文系根据作者应邀在 1989 年 6 月 14 日日本情报产业协会(JISA)与软件工程学会(SEA)联合举办的 1989 年软件年会(Software Symposium'89)上所作的题为"XYZ 环境"基调讲演中部分内容整理扩充而成. 因该会议论文集中所收入的介绍 XYZ 系统文章[1]未写入这部分内容, 故特写成本文作为[1]的补充.

XYZ 系统是一基于时序逻辑且适应多种程序设计方式的 CASE 环境. 它所适应的程序包括过程性的, 抽象描述型的, 或产生式规则型的; 串型的或并发的; 常见高级语言型的, 面向多种对象的, 函数式的或逻辑的; 文本的或图形的, 后者又可以是数据流型的或控制流型的等等. 整个系统由经纬两维组成: 一维是一具有统一程序框架的时序逻辑语言 XYZ / E. 它是整个系统的核心和经络, 由许多子语言组成, 以适应多种程序设计方式, 而且它还可用来表示常见高级语言, 某些专用的抽象描述语言(如面向分布式进程通信的

* 1989年7月21日收到.

© 中国科学院软件研究所 http://www.jos.org.cn

描述语言 Lotos)，以及多种图形语言(如数据流图 DFD 以及由 Petri 网扩充而成的控制流图 CFC)等的操作语义．XYZ 系统的另一维是由多种协调的软件工具或 CASE 环境组成．由它们既可实现由常见高级语言或专用描述语言到 XYZ／E 的可执行子集的形式转换(再加上由属性文法表示其静态语义部分而成为一建立在形式语义上的编译的编译系统)；亦可支撑由需求、抽象描述、转换、验证或快速示范等步骤组成的程序设计方法论．其中转换可以是逐步分解型的(decompositional)，也可以是逐步作出决定(decision)，然后以由非形式到形式，由抽象描述到有效实现的方式进行过渡的方式．这些方法既可用于文本程序的设计，亦可用于 CFC 或 DFD 图形程序的设计．在后一情形下还可自动生成由时序逻辑语言 XYZ／E 表示的操作语义文本，并证明前后步骤所产生图形的语义的一致性．与这些方法相应，系统中还包含一统一的可重用对象库以存储程序开发过程中每一步所产生的信息，这些信息既可用来追索程序设计过程，以帮助用户对它的描述的理解和验证，也可用作修改或重用这些描述的参考．这系统是在 Sun 工作站上实现的，它是在 Unix 与 X-Window 上新加的一层通用的基于形式化方法的软件开发手段，用户可用它们开发自己的具有精确语义的专用软件．

XYZ 系统的设计思想，概括起来可分为二个方面：

(Ⅰ)试图在理性主义与实用主义之间找到一种合理的平衡；

(Ⅱ)试图将近年出现的各种有意义的程序范型，在一协调的形式框架中统一起来．

作为(Ⅰ)的直接后果，统一地以时序逻辑语言表示系统中提供的工具的精确语义界面，以及由系统提供的环境所生成的程序的操作语义，再加上可重用对象库的图式管理系统对程序设计过程所产生的信息进行管理，这样就可大大加强程序的可重用性与可适应性．

作为(Ⅱ)的直接后果，由于由 XYZ／E 表示的过程性程序，抽象描述及产生式规则具有统一形式框架与计算模型，致使不同范型的程序块可在 XYZ 系统中任意组合并协调地运行．因此，XYZ 系统可方便地支撑基于知识的程序设计以及其它多种范型混合的程序设计．

下面就(Ⅰ)与(Ⅱ)作进一步的说明．在作此说明前，先阐述两方面共同的方法论哲学基础．

§2. 分与合的哲学

我在大学读书时曾学过哲学．当时最流行的西方哲学流派为著名的剑桥学派[4]，其代表人物有罗素(B.Russell)，摩尔(G.E.Moore)以及稍后的维特根斯坦(L.Wittgenstein)．这一学派又名分析哲学，其方法论主要特点是对概念作逻辑分析．我现在已不大记得清这些哲学家著作的某些具体内容，但他们所提倡的分析方法却一直在我所从事的学术工作中起作用．我发现许多表面上矛盾的概念，如对之进行深层分析，有时可能变成相容．比如，常见程序语言中的变量具有动态取值的特征，也就是在程序运行中可以新值覆盖旧值；而逻辑系统中的变量概念则是静态的，其值是一次决定的．故两种变量概念是矛盾的．但如采用 Lucid 模型[5]，以可随时间延伸的向量解释变量，则上述两种不同的变量概念即可以统一起来．以此作为基础，即可用一致的逻辑框架将直言式的功能描述与命令式的状态转

© 中国科学院软件研究所 http://www.jos.org.cn

换以统一的方式予以表示，从而填平逻辑与程序语言之间的鸿沟.

从五十年代起，我又从毛泽东的哲学著作中学到另一种分析方法．这就是对具体矛盾进行具体分析的方法．其基本点是：从事物的发展变化过程中，历史地对事物内包含的具体矛盾进行具体分析．我深感到这方法使用得当，对发现事物的发展变化趋势，分清问题中各方面的主次，提出解决问题的办法很有帮助．我在文[2]中即试图应用这种方法对程序技术发展三十年的历史与现状进行分析．文[2]指出，不论从技术方面说，还是从理论方面说，程序技术研究在七十年代的特征是由单一走向分散，出现了许多新的技术领域，许多不同的方法与互不相容的理论体系；而到八十年代则与这种"由合走向分"的动向相反，是"由分走向合"，即将各种分散的技术与理论综合起来组成水平更先进的工程性系统．具体内容已在[2]中详述，此处不赘，下面只想从哲学的角度来谈谈．

这种具体分析矛盾的方法有以下哲学前提：(1)事物发展变化是客观存在的，不依赖于人们的主观愿望．但人们如对之有清醒的认识，则可因对其产生变化的内在因素或其中的某些方面，施加影响从而引导变化的方向与程度，(2)产生变化的因素有事物内在的，也有外在的，而前者是起主要作用的．(3)这种促使事物变化的因素即矛盾．矛盾各方有主有次，这种主次地位是可以在过程中转化的，同时，矛看的各方在一定条件下可以融合，也可以引起冲突．正是由于这些转化、融合与冲突，构成事物的发展变化．

下面我们进一步看看，事物是以什么样的方式进行发展变化的．对此，可以有三种不同的哲学观点：(a)认为事物变化过程中只有量的变化，无质的飞跃．这是所谓进化论观点．这种观点如引向极端，则是一种不承认革命的改良主义．在 Stars 计划中，"进化论"观点属于此类．(b)另一极端即认为事物变化过程是由接连不断的质变所组成，不承认质变之前量变阶段的意义．这是一种急躁的革命论，容易产生冒进盲动，终必引起巨大的阻力而失败．在 Stars 计划中有一种"革命性"观点，如将这观点推向极端即可产生这种后果．程序语言或方法论均与人的思维方式相关，要改变人们已习惯的思维方式是很困难的事，必然遭到习惯的阻力．形式抽象描述与人们久已习惯的高级语言编程方式完全不同，企图在短时间内要求人们放弃已习惯的后者而采用前者显然是不易被接受的．欧洲一批力量很强的专家希望在工业界推行指称语义方法已近十年，至今仍然收效甚微即说明了这一点*．(c)显然，较妥善的态度应该是既承认量变也承认质变．质变只是在原有质的基础上进行长期量变，逐步将新的质引入并融合于旧的基础之中，在积累到充分程度以后才进行的飞跃．只有这样的逐步变化方式才可促使事物顺利的发展．虽然这一原则的正确性易为人们所理解，但具体实施远非易事，关键是既要克服主观方面的片面性，又要能实际在事物发展变化中根据具体条件不断掌握适当的分寸，并准确地分清界限．而且，在旧质基础上引入新质时，如何能使矛盾的方面作合理的安排使之协调，这就很需要进行精微的分析，分清层次与范围，绝非简单的混合．这样形成的过渡性对象，既是新旧质的融合体，必然"非驴非马"，从哪一个方面看可能都不是完美无缺的，不是很彻底纯净的．而根据当时条件从整体发展看却可能是最佳选择．如何看待或评价这种中间发展阶段形成的对象，涉及哲学方法与价值标准问题．这个问题不解决，就很难在实际事物发展过程中处理量变阶段的矛盾．我认为，一种值得推荐的哲学方法是中国古代儒家的"中庸之道"或辩证法的

* 最近丹麦的Bjorner教授与本文作者面谈时，亦承认了这一事实.

© 中国科学院软件研究所 http://www.jos.org.cn

"合二而一"理论．一对矛盾存在于统一体中，必存在其对立的方面，亦必存在其统一的方面．无前者而不成其为矛盾，无后者则彼此无联系亦无所谓矛盾．在质的飞跃阶段，应强调矛盾的对立方面，不然就不能促成突变*．在量的变化阶段，却应强调其统一的方面，否则即不能进行融合．朱熹对"中庸"的解释是："中者不偏不倚、无过不及之名，庸，平常也"．程颐认为，中庸的理论"始言一理，中散为万物，末复合为一理"．由此可见，中庸并不是简单的折中主义，而是告诉人们如何处理由分到合再由合到分的方法．而且在融合矛盾的两方面时，应从实际出发注意防止片面，避免偏激，掌握分寸，做到按事物的常理将对立的方面安排得恰如其分．而且这种安排应根据当时发展的具体条件来选择，所以说"君子而时中"．这是一种处理实际问题的有效方法，而不是评价理论，建立学科体系的方法．我认为，在程序技术发展的现阶段，需要强调由分走向合，将各种分散对立的技术和理论进行选取的综合，组成水平更高的软件工程环境．在这样的时期，强调一下"合二而一"，注意一下调和矛盾的"中庸之道"，无疑是有现实意义的．

XYZ系统设计思想的(Ⅰ)与(Ⅱ)两方面，正是根据程序技术发展过程中产生的以下两方面的现象提出的：(i)软件工程技术与形式化理论的分离与脱节以及语义理论本身的分散发展，(ii)程序范型的分裂．由于我们认识到现阶段的程序研究的总趋势是由分走向合，所以，在具体分析了以上各方面矛盾的基础上，根据我国以及世界学术发展现阶段的条件，试图应用"合二而一"及"中庸之道"的方法，将矛盾的各方面进行了精心的剪裁与安排，组成了这一系统．应该承认，(a)这种集成的道路不是唯一的，XYZ系统也不能说从各种角度看都是最优的选择，但我们认为XYZ系统不是草率完成的工作，至少在设计上是经过了反复推敲研究的，经过慎重权衡与修改的．(b)既然这只是一种程序技术发展量变阶段的工作，所以从任一局部的理论或技术看，都不能说有惊人的创造．甚至，有时为了全局的需要，我们还宁可牺牲某些局部的新奇与完美．为了全面权衡得失，不可避免地只能采取"中庸"的态度．比如，现阶段作为XYZ系统核心的时序逻辑语言XYZ/E，选择的是一阶时序逻辑系统．这样的逻辑有许多优越之处，用户较易接受，也较易实现，而且它的表示能力已相当的丰富，常见的程序概念多能用它描述．但是，有些深层的程序概念，它却无能为力，比如形式过程参量，受囿局部变量等均要求高阶逻辑系统．但为了保持系统的简明性，易于被接受和工程实现．我们只好分成二步走，目前面向广大用户的系统采用一阶系统，至于高阶逻辑，则留待以后扩充以面向少数特殊用户的需要．这些选择标准都带有很重的工程特征，也就是我们所谓的"中庸之道"．下面，我再应用这样的哲学方法，对前述(Ⅰ)与(Ⅱ)两方面作一些说明．

§3. 理性主义还是实用主义

前几年，我多次访问了北美与西欧．我得到的印象是，虽然这些地区的国家在深刻的形式语义理论及先进的软件工程技术各方面都有很好的工作，但从总的风气讲，两地区所强调的侧重面是不同的．西欧国家更强调形式语义理论的研究以及计算机科学的学科基础

---

* 我个人体会，毛泽东曾将他的方法论概括为"一分为二"，即旨在强调矛盾的对立性这一方面．在变化的质的飞跃的阶段作这样的解释是合适的．

© 中国科学院软件研究所 http://www.jos.org.cn

的建立，而北美的技术界则更看重新软件技术的创新及其应用．应该承认，这两个方面对形成更高水平软件都是十分重要的．在七十年代，为了解决软件开发中产生的困难问题，在一定时间内分开进行某一方面单刀直入的深入探索，不同地区根据自己的条件各有侧重地进行研究，可能是必要的，大有益于最后促进软件技术发展的．可是，到了现在的发展阶段，总的潮流已趋向于综合各方面的成果组成更先进的系统的时候，如果仍然以这种南辕北辙的方式片面强调一种价值标准，促使理论与技术向各执一端的方向发展，则势必使理论与技术日益分离，作茧自缚，最终起阻碍计算机科学与技术向健康方向发展的作用．这就是我所感到的理性主义与实用主义的矛盾．这种矛盾在一定情况下之所以有害，主要是由于脱离了当前技术发展的潮流，每一方片面强调所习惯的一种哲学价值标准而忽视了另一方的意义所致．事实上，世界上早已有许多重要的计算机科学家指出了这方面的问题．斯坦福大学的 Knuth 教授早在七十年代中期即曾在数月内在欧洲举行的两次国际会议(世界数学会及国际控制论会议)上作过两个报告，前一报告强调数学在促进计算机科学发展中的重要作用，后一报告则着重说明不适当地强调将研究数学的方法和标准用于计算技术的研究与开发，有可能起的消极影响．最近将在旧金山召开的国际计算机会上，他又将作题为"理论与实践"的基调讲演说明二者的相互影响，从而强调两方面结合的重要意义．应该说，这是一次很及时的箴言！事实上，近数年来，在软件工程领域正出现一种新的趋势，即将软件工程方法、工具与环境方面的新技术与形式化语义理论有机地结合起来，形成更高水平的 CASE 系统．这就是所谓"建立在形式化基础上软件工程环境"(Formally Based SE Environment)[6]或"建立在语义基础上的程序设计环境"(Semantically Based Program-Design Environment)[8]．这一趋势的出现标志着程序技术发展进入一新的时期，即理论与技术有机结合的时期．它一方面表明形式语义理论研究日益成熟，已进入工程实用的阶段；另一方面也表明，软件工程技术已发展到这样一个地步，它要求使其复杂的工具具有精确的语义界面才便于安全地使用，同时由它所支撑开发的程序块，具有精确的语义，才可以合理地重用．下面，Brown 大学 Aia Giacalone 等的观点颇具代表性[7]．

"一程序设计环境中如提供建立在形式化基础上的工具与运算作为手段以在统一的形式框架中操纵程序，将比不具这种特色的环境有重要的优越性，特别是在这些工具及由它产生的程序的一致性，概念的简明与协调等方面．

概念的协调性在为用户提供含义精确的工具与运算，使他们能根据自己的特殊需要将基本环境予以扩充方面具有本质的重要性．如果这种工具能在形式化层次上予以开发，从而能被精确地理解，它就更能向用户提供关于其行为的非形式化的或半形式化的，然而却是精确的定义．"

形式化理论与软件工程技术结合虽是应受欢迎的好事，但并不是简单易行的．因为两方面各有自己的价值标准．遇到具体结合的情形，即可能出现矛盾．形式理论工作者往往看重概念的精美，表示能力的强大，及系统的完整性等．而软件工程专家则更注意用户方便性，时空效率，系统可靠性，程序可重用性及可适应性等．如果任何一方执着于自己所习惯的一种标准，则遇到矛盾时很难协调．事实上，要使双方结合组成系统，势必针对具体问题各有取舍，最后得到的系统从那一方的标准看都有不能尽如人意之处，但只要从总体看是一最合理的选择即可．这里，就不免要让"中庸之道"发挥作用．这事实上是处理实

© 中国科学院软件研究所 http://www.jos.org.cn

际问题不可避免的现实主义态度.

目前已存在的形式语义理论很多，究竟应如何选择？事实上，已存在的各种理论均有其适应的方面与不适应的方面，没有一种理论具有全面的优越性．因此，当前流行的做法是选择几种方法予以适当的剪裁、组合，使之在一协调的系统中各自起合适的那一方面的作用．当然，这并不是说，在一系统中各种理论起同等的作用．特别是一系统要求用户熟习所有各种形式理论才能使用，必然遭到用户的拒绝．所以，在一系统中必有些理论是起主要作用的，是用户需要了解的，另一些理论只能起次要的局部的作用，用户可以忽略．

现在比较流行的形式化方法及描述语言，大致可分以下三类:

(A)基于代数映射的形式方法．这里可包括指称语义、代数语义、带类型λ演算等．直觉主义逻辑及构造主义证明方法亦可归入此类．这些方法的优点是数学形式较为精美，理论基础深厚．在欧洲学院式研究中占压倒优势，可以说是计算领域中理性定义的代表．但从工程实际的角度看，则有许多不足之处．(i)与长期软件工程技术界编制程序的习惯相距太大，(ii)要求使用者有较好的抽象数学训练，(iii)许多重要的程序概念和性质如并发进程，特别是活性，很难用此方法自然地表示，(iv)实现的基础多要归结为函数式语言，这种语言的执行效率及表示能力均有一定的局限性．因此，这一方向虽经大量高水平的专家长期努力，仍难以为工程界广泛接受．不过在某些特殊的专用领域，如CCS在在通信协议领域，以及代数公理方法用于描述基本数据类型等方面，这种方法仍是很有意义的．

(B)基于编译技术的形式方法，比如属性文法及扩充的有穷自动机理论等．由于后者目前尚未构成一完整的体系，此处很难评价．至于属性文法则是目前最能为软件工程界接受的形式化方法之一．它是将语法分析方法在上下文相关部分直接扩充而成，紧密与编译技术(特别是名字表技术)相联系．所以，这方法具有理论与实践密切结合的特征．但它也有些不足之处，(i)由于直接反映编译技术的要求，对于一些不太熟习编译的内部结构的人，往往感到难以理解和书写，(ii)抽象性不够，反映技术细节太多，(iii)较适于表示静态语义，对于实际执行过程的操作语义，显得无能为力．所以，这一方法，被许多系统(Ergos, XYZ)采用来表示静态检查．

(C)基于逻辑的形式方法．包括古典逻辑及时序逻辑等．由于古典逻辑可看作是时序逻辑的特殊情形，甚至Prolog这种建立在Horn子句上的系统亦被许多时序逻辑系统(如XYZ／E，及Temporal Prolog)所包括，所以，这里只需考虑时序逻辑就够了．由于XYZ系统采用时序逻辑为基础，所以，下面更着重地说明一下时序逻辑的优缺点．1983年IFIP'83会议上L.Lamport对时序的优越性作了以下的概述[9]:

(i)"时序逻辑是古典逻辑的一种颇为简单的扩充．它的确提供了描述一程序的时序行为的一种自然的途径".

(ii)"时序逻辑是适于对并发程序进行抽象描述及论证的好方法．…这种时序逻辑允许你表示出一程序的两方面性质:

· 安全性，即肯定一程序不会做坏事.

· 活性，即肯定一程序最终将做好事."

(iii)"虽然有许多理由说明我为什么喜欢时序逻辑，但有一个理由是主要的：时序逻辑以一种简明而自然的方式支持分层描述与论证．经验表明，描述一复杂系统的最佳途径是通过分层的抽象，从高层抽象描述起直到用一程序语言实现止，每一层是下一层的抽象描

© 中国科学院软件研究所 http://www.jos.org.cn

述，同时又是上一层的实现，时序逻辑提供一统一的逻辑系统描述各层次的抽象——从最高层次的抽象一直到用程序语言实现."

(iv)"时序逻辑是形式逻辑的一个分支，它对许多人还是一新的理论．当一新的形式符号系统提出时，人们常常宣称它是简明的，自然的，而且容易学的．…但学习一新的形式符号系统，绝不是件易事，因为它要求学会用新的方式思维．你可能发现，在开始学谓词演算中的 与 时也是很困难的，只有当你学会改变你的思维方式时，这形式符号才变成简单而自然的，学习用时序逻辑对程序进行论证，其难度与学着用 与 十分相似."

XYZ/E是建立在Manna-Pnueli的线性时间时序逻辑理论[10]基础上的语言[3][1]．它在这逻辑系统之上引进一种统一的程序单元图式．这语言除了具有上述L.Lamport指出的特征外，还由于引进了表示赋值及控制转移的等式，使表示算法的状态转换命令与表示功能的直言描述可以用统一的形式表示出来构成统一的计算模型．这是最早提出的可执行时序逻辑系统之一．它具有统一范型的特色．

当然用时序逻辑作描述工具也不是十全十美，其主要的常被指出的不足之处，也就是逻辑理论所共有的以下两方面的缺点："使用逻辑形式体系存在两个基本障碍．其一，一般逻辑系统中最令人感兴趣的任务(例如检测矛盾)是计算上难以处理的．…其二，对于大多数人来说，复杂的逻辑公式特别难于书写和理解"[11]．

为了避免这两方面的问题，我们认为最有效的途径不是去寻找新的或修改旧的形式符号体系，而是求助于进行描述或使用描述的程序设计过程．显然，只有复杂的多层嵌套的时序逻辑描述才会发生难于书写，理解与验证或查错的问题．对于这样的复杂问题的设计，我们在XYZ系统提供了一组工具与环境，它支撑一种可称为"根据用户逐步作的决定进行描述、转换与验证(或快速示范)"的方法论．从[8]中可看出Ergos计划也采取了极为相似的途径．根据这一方法，一复杂程序的设计是由一序列的步骤组成，每一步中用户根据问题的需要分清轻重缓急作出决定，并将这决定由非形式描述逐步转换成形式描述，由一图式管理系统将所有这些信息存入可重用对象库中以备以后查询，然后验证或查证这一步作出的形式描述是否与上一步的形式描述保持一致．每一步均按这样的方式进行直到得出可有效执行的程序为止．这一方法论的优点在于将一复杂程序设计过程由一次性描述与验证变成一逐步进行描述修改和验证的过程，而且将每次为新描述作的决定的有关信息均保存在一可重用对象库中以备查找，由此用户即可了解每一步所作决定的意图，从而使由此作出的描述较容易理解和验证．这样即可使这困难的问题变得容易．一次性描述之所以令人难以理解与验证其根本原因是所有描述过程中与作决定有关的信息均已丢失或掩藏，致使阅读最后的描述文本令人如坠五里雾中．上面这一方法无非是恢复了设计过程的本来面目而已．这样一方法论及其支撑环境应该说是有明显的优越性的，但要广泛为工程界理解与接受恐怕还得需要时间，且还应增加自动化或半自动化的辅助工具及文档手段．此外，恐怕还需要写出符合工程界口味的教程并进行培训．

Ergos与XYZ系统都采用了上述程序设计方法并提供了相应的支撑环境与工具．但二者的形式化基础是不同的，XYZ是以时序逻辑语言为基础，Ergos则以带类型的λ演算等为基础，两个系统都以属性文法作为描述静态语义的方法．

以上是就通用的描述语言及支撑工具而言．至于专用领域，其描述语言可根据该领域的特殊性而定，比如Lotos这种专用于描述通信协议的语言，则是以代数方法为依据的．

© 中国科学院软件研究所 http://www.jos.org.cn

这样的专用系统与上述通用系统是可以结合起来发挥作用的．前面已指出，在 XYZ 系统中提供了一基于形式语义的编译的编译系统，以属性文法表示静态语义，以一种形式化元语言表示由所给源语言到 XYZ／E 的形式转换，通过转换得到的 XYZ／E 程序表示源程序的操作语义．这系统既可用于专用描述语言亦可用于常见高级语言到 XYZ／E 的转换．

我们试图通过上述途径在理性主义与实用主义之间保持一种合理的平衡．实践将表明这目标是否已达到．

§4. 范型统一问题

在七十年代末之前，计算机体系一直统一于 Von Neumann 型，而程序范型则统一于以这种体系为基础的算法语言．经过七十年代计算技术与理论的分散发展，这种统一性被打破了，出现了许多常见高级语言以外的程序范型，如函数式程序设计，逻辑程序设计，基于抽象描述的程序设计，产生式系统，面向对象程序设计，图形程序设计等．它们所反映的计算模式都超出了 Von Neumann 体系的范围．而这些程序设计方式各有其适应的范围，任何一种也不能取代其他而独占统治地位．

这种范型分裂的局面也同时给程序设计带来不便．比如抽象描述与可有效执行的算法程序之间如果存在不可逾越的鸿沟，则由前者向后者过渡时将难以保证语义的一致性，同时在过渡的中间阶段，不可避免将出现两类程序混合出现的情况，这样的程序应如何表示？此外，近年来基于知识的程序设计很为人注意，如果算法与知识由两类不同的语言表示，二者如何渗合沟通？因此，很自然地产生了范型统一化的要求．UT Ausfin 的 Chandy 与 Misra 在[12]中的一段话与 XYZ 设计思想很相近似：

“今天，程序设计好像日益被分裂成许多神秘的支派，每一支有它们的传教士、僧侣与符咒．我们相信，在所有各类程序设计的基础之下，存在一种统一性．我们愿为识别它而作出贡献．”

XYZ 系统正是以此为主要目标之一而设计的．我们认为，这种统一性就是各种类型程序的语义模型．而且相信，时序逻辑语言 XYZ／E，正好可以用一种一致的形式框架将这统一的计算模型表示出来．这一致的逻辑框架，既可用来表示抽象描述，亦可用来表示有效算法，还可用来表示产生式系统；不论是串型的或并发的，决定性的或非决定性的程序都可在这逻辑框架中得到自然的表示．

有了这种统一的程序框架，则在基于抽象描述的程序设计过程中，从最抽象的描述到可有效执行的算法之间各种层次的程序以及中间过渡的混合型程序均可以统一的方式予以表示和相互连接，从而可进行分层程序设计并验证上下层程序之间语义的一致性．

有了这种统一的程序框架，则在基于知识的程序设计中，不论算法程序或表示知识的规则及其查询命令，也可以统一的方式得到表示，而且可以协调运行，互相通信．

有了这种统一的程序框架，则在面向对象的程序设计中，可以面向多种类型的对象，而且可以用统一的方式表示这些不同类型对象的语义．事实上，只有建立了这种统一性的条件下，面向多种对象的系统才是真正有意义的，因为此时不同类型的对象之间才能有“共同的语言”进行语义沟通．

© 中国科学院软件研究所 http://www.jos.org.cn

有了这种统一的程序框架，则许多在软件工程中有用的图形语言，如DFD，Petri网等即可以找到一种方法表示出它们的操作语义，从而使这类语言结点进行分解并验证其语义一致性变得颇为自然和方便。

目前，XYZ系统还只实现一可运行的试验性系统，还有待进一步的改进使之工程化成为一完善的系统。我们相信，上述这些设计目标最终将可以达到。

感谢： 本文作者对这次日本软件年会的东道主藤野晃延先生及SRA的岸田孝一先生及会议的组织者和SRA的同行们对本人及我的助手的盛情邀请与接待，以及为进行讲演与演示所给与的多方面的支持与帮助，表示由衷的谢意。

参 考 文 献

[1] 唐稚松，"XYZ环境"，ソフトウエア・シンポジウム'89论文集，1989年6月14–15日，东京。

[2] 唐稚松，"程序技术研究三十年"，计算机科学，1988，第3期。

[3] C. S. Tang, "Toward a Unified Logic Basis of Programming Languages", Proc. IFIP'83, 1983, Paris.

[4] 约翰·查尔斯沃斯著，田晓春译，《哲学的还原》，四川人民出版社，1987。

[5] Aschcraft E. A. and W. W. Wadge. "Lucid: A Nonprocedural Language With Iteration", Comm. ACM 20, 1977.

[6] Giacalone A. et al., "Toward a Formally-Based Programming Environment", 《Integrated Interactive Computing Systems》, (ed. P. Dgona & E. Sandewall), 1983.

[7] Cleaveland R. et al., "The Concurrency Workbench: Operating Instructions", Tech-Note 2/88, Computer Science University of Sussex GB, 1988.

[8] Lee P. et al., "Research on Semantically-Based Program-Design Environments: The Ergos Project in 1988", Tech. Rep. No. CMU-CS-88-118, Computer Science Dept., Carnegie-Mellon Univ., 1988.

[9] L. Lamport, "What Good Is Temporal Logic", Proc. IFIP'83, Paris, 1983.

[10] Manna Z. and A. Pnueli, "Verification of Concurrent Programs pt. Ⅰ-Ⅱ", Dept. Comp. Sci., Stanford Univ., Stan-CS-81-836, Stan-CS-81-843, 1981.

[11] Rich C. and R. C. Water, "Automatic Programming: Myths and Prospects", 中译文，计算机科学，1989，No.3.

[12] Chandy K. M. and J. Misra, "Parallel Program Design: A Foundation PTI", Draft, Dept. of Comp. Sci., Univ. of Texas, Austin, 1986.

© 中国科学院软件研究所 http://www.jos.org.cn

第九届全国数据库学术会议(1990.9.10–14)

征文通知

经中国计算机学会软件专业委员会同意，由中国数据库专业学组委托上海第二工业大学、公安部第三研究所(上海)、复旦大学、华东计算技术研究所举办第九届全国数据库学术会议。会议定于 1990 年 9 月 10 日至 14 日在上海召开。这次会议是我国数据库学术界及数据库工程与应用界又一次盛大的会议。特发会议征文通知。

征文内容

1.数据模型与语言；2.数据结构与存贮方法；3.数据库设计方法与工具；4.数据库工程与应用；5.数据库理论与算法；6.用户接口；7.分布式数据库；8.知识库系统；9.多介质数据库；10.面向对象数据库／知识库。

重要日期

征文截稿日期：1990.3.30　　发出录取论文通知日期：1990.4.25

审稿日期：1990.4.15　　交出版社日期：1990.5.20

提交论文要求

由于会议拟出论文集，请严格按照会议重要日期要求并对论文提出下列具体要求：

1.每篇论文字数不超过 8000 字，短文 4000 字(均含图表、附录、参考文献等)；2.论文书写次序：标题、作者姓名、单位、摘要、正文、参考文献、英文摘要；3.正文分段小标题序号用阿拉伯数字，1，2…；4.论文中文摘要 200 字，含关键字(另起行)；5.论文提交一式二份，自留底稿，不再退稿。稿件图形、文字清晰；6.恕不受理已在杂志上或其他会议上发表的文章。

组织机构

程序委员会

主　席：萨师煊(中国人民大学)

副主席：罗晓沛(中国科技大学研究生院)；施伯乐(复旦大学)

组织委员会

主　席：何守才(上海第二工业大学)

副主席：尹良滨(公安部第三研究所(上海))；娄荣生(复旦大学)；瞿兆荣(华东计算技术研究所)

稿件投寄：

200041　　上海陕西北路 80 号

上海第二工业大学计算机系

第九届全国数据库会议　　会 务 组　　朱　宁　　收

(请填写寄稿人地址、邮码、姓名。)

© 中国科学院软件研究所 http://www.jos.org.cn