计算机科学 1996 Vol. 23 №. 2

# 可视语言十年进展

张自力
(西南师范大学计算机科学系 重庆 630715)

**摘 要** 本文对可视语言研究十年来的历史进行了简要的回顾，并着重对可视引喻(Visual metaphors)，可视语言中的形式方法和可视程序设计等方面的研究进展进行了较详尽的综述。

**关键词** 可视语言，可视程序设计，可视引喻，可视化，人机接口。

自1984年在日本召开第一届 IEEE Workshop on Visual Languages(VL)以来，可视语言正式成为计算机科学的一个相对独立的研究领域，正日益受到人们的广泛重视。据权威人士预测，在本世纪剩下的几年里，硬件方面主要发展方向是多处理器和多媒体(Multimedia)，软件设计的主要方法是面向对象(O-O)，可视语言正好顺应这一潮流，可以充分利用其成果，并把它们推向更广泛的应用，与之相得益彰[1,2]。为使对 VL 研究感兴趣的读者对这一新兴研究领域有一个较为全面的了解，本文对 VL 十年来的研究历史进行了简要的回顾，并着重对可视引喻，可视语言中的形式方法和可视程序设计等方面的研究进展进行了较详尽的综述。

## 1 可视语言研究回顾

可视语言来源于图形界面设计和程序可视化(Program Visualization)。可视语言是大量依赖图形或非文本成分作为其通讯能力的语言，通常具有两层含义：其一，该语言处理的对象是可视的——用于可视信息处理的语言即可视信息处理语言；其二，该语言本身是可视的——具有可视表达的，用于程序设计的语言即可视程序设计语言。

目前，有关可视语言方面研究的文献主要出现于可视语言国际会议论文集(Visual Languages Proceedings，从1986年开始每年一次)及 Journal of Visual Languages and Computing(创刊于1990年3月)。由于许多参考文献都出于这两者，文末不再附这些文献，而只在文中相应地方标记：(作者名，VL'nn)或作者名(VL'nn)及 JVLC(年、卷、期)。十年来，从收入 VL 论文集的文章来看，约85%的文章落在可视程序设计和语言，非计算问题(noncomputing problems)方面的应用及理论等三个主要方面，在1993年的 VL 论文集中，这三方面的论文占了近95%，而在这三方面中，可视程序设计语言的定义和使用方面的论文又占了40～60%。

许多可视程序设计语言均把重点放在面向对象的程序设计，从而使程序设计变成通过图形对象重叠、并列或互联等方式进行的对象交互的表示。可视程序设计语言中的另一论题是算法动画制作(Algorithm animation)，其目的是表示程序中的某些方面如数据或控制流的动态变化，但褒贬不一。

人机接口问题与可视程序设计语言及应用领域都密切相关。接口所表现出的可视引喻既要便于用户访问，又要便于程序设计系统解释。十年来在人机接口方面的研究主要包括从 ICON 设计，到屏幕格式化，到鼠标及其它与可视屏幕图形进行交互的方法的研究。

可视语言的理论研究进展缓慢，由于其二维(甚至三维)的特点，使其与自然语言和绝大多的形式语言不同。这十年间，有许多论文讨论可视语言的一般性质。理论研究方面的一个良好开端是在 ICON 化语言(即语言的基本组成是 ICON 而非文字)方面，在这方面，S. K. Chang 所领导的小组做了大量的工作。

另外，可视化(特别是科学可视化)，直接处理图象的数据库和虚拟现实是已引起高度重视的，与可视语言密切相关的研究领域。

## 2 可视引喻进展

引喻本是语言学中的一种修辞手法，在这里，主要是指用 ICON 作为过程或对象的可视替代物或作为可视语言的原语(primitives)。一个 ICON 可定义为"一个能告诉观察者关于由设计者赋予的内在信息(概念、功能、状态、方式等)的可视的被分割的对

张自力 副教授，主要研究专家系统、近似推理、多值逻辑等。

象"，或"一个可感知的物理模式，一个用户群体同意赋予一定的意义，便于人们相互间的通讯和推理"。这后一个定义更强调用户对ICON解释的重要性，同时也指出要形式化描述一个具有可视引喻的系统，需要有四个层次，即词法、语法、语义和语用。

利用可视引喻可以方便地表示许多操作或行为，如利用文件ICON可使添加、合并、排序、打印等操作一目了然，从而避免了用文本语言来描述相应行为时存在的各种限制(传统程序设计系统固有的线性性便是限制之一)。

可视引喻已经或可望在以下五个方面获得广泛的应用：1)可视程序设计；2)数据库查询；3)设备维护与故障检测；4)数据解释和5)用于协作工作的系统(Systems for collaborative work)。

### 2.1 可视程序设计方面的工作

Kramer(VL'84)最早提出了设计一种用于建造可视用户接口的方便工具的方法，其核心是采用面向对象的程序设计方法来创建新的ICON类。

Tanimto(VL'86)提出了一种用于程序设计的易于学习的可视语言，其重点集中在问题的表示，即找出数据、程序、过程和命令对象的好的可视计算引喻。

Haarslev(VL'88)提出了一种基于数据流范例的可视语言。该语言可用于科学计算和动态过程的可视化。在该语言中引入了功能单元的概念并定义了功能、模块、目标、程序和过程概念的合一。一个功能单元(FU)可看作是由一组参数($P_i$)控制下的输入数据项($I_i$)到输出数据项($O_i$)的映射：

$$FU(P_1,\cdots,P_n):(I_1,\cdots,I_n)\rightarrow(Q_1,\cdots,Q_n)$$

Beretia(VL'86)讨论了在用于图象解释的可视语言中，将ICON作为原语来综合复杂的思想，并提供了强调人与人之间可视通讯重要性的许多定义。ICON被视为已分类的对象，一旦这些ICON被人解释，便可选择用于完成某些动作，如"显示某些数据"等这样的动作。

*Lingua Graphica*(LG)(Stiles VL'92)是一个用于虚拟环境的三维可视语言。该语言利用三维立体语言构成来建造程序和系统脚本并将其可视化。设计LG的目的是使得能够利用二维可视及三维可视的程序来创造有用的虚拟环境。

Lee(VL'92)讨论了可视翻译问题，可视翻译定义为能够将口语译成符号语言的一个程序。相反的工作，即将可视符号语言进行解码以便译成文本语言的模式识别也有人探讨。

### 2.2 数据库访问

数据库访问是一个非常重要的领域。事实上，已有许多工作试图利用可视查询语言使数据库访问变得方便快捷[1]。在这方面，最早的工作是Hirakawa(VL'87)引入的用于受限域数据库的可视查询，更进一步的工作已经扩展到了ICON(来源于一般化ICON理论[2])特性。

Tsuda(VL'89)设计了一个用于从基于ICON(ICON既代表类又代表数据库中的对象)的面向对象数据库中检索信息的导航系统。在该系统中，查询可以通过重叠代表数据库中对象的目标ICON来形成。

### 2.3 设备维护和故障检测

在设备维护和故障检测领域，Cinque(VL'87)提供了一个雷达设备故障检测的解决办法。它利用一个ICON系统，将可能的各种错误配置显示在雷达的CRT上。为便于非专业人员维修复杂的电子线路故障，与该ICON系统相联的还有一个专家系统。

Mears(VL'88)利用可视引喻设计了一个用于培训推进器工程师的计算机模拟系统。在该系统中，图形对象(ICON)既用于表示又用于交互。

除此之外，在一个称为彩排世界(Rehearsal World)的系统中引入了环境引喻，程序设计活动被舞台戏的趋向所表示。

### 2.4 数据解释

在许多数值计算应用(如科学模拟)中，一般都要产生大量的数据。在数据解释领域，Williams(VL'89)设计了一个系统EXVIS，利用基本ICON(条形物体)来表示科学数据。每一ICON可具有一给定值的分支(Limb)。一个ICON图形显示的纹理中有趣的梯度或等高线可能指示了基础数据中的重要性质。

### 2.5 协同工作的系统

最后一重要的领域便是协作工作的系统。在这样的系统中，人员构成的网必须进行富有成果的交互以达到一个共同的目标。

Swenson(VL'93)引入了一种可以描述和支持协作工作的规划与在线的可视语言。协作可在规划阶段完成。计划与政策(预先确立的计划)可利用可视过程语言VPL(*Visual Process Language*)来创建与编辑。协作软件的目的在于协调一个组织中不同成员的活动，让每一个成员了解一个小组或组织正干什么及支持什么样的过程改变。

利用引喻的匹配性质(源与目标领域的相似性)和错配，引喻可用于接口设计。Levialdi(VL'93)从作为网证的观点分析了现有的可视接口，仍产生了ICON的二元定义，即一个ICON由物理部分(图象)和逻辑部分(意义)构成。

交互作用怎样才能恰当地形式化以被正确理解并用作下一代利用可视引喻的系统的框架呢?这还需继续深入的研究。

## 3 可视语言中的形式方法

可视语言的理论提供了一个理解可视语言的理论基础,因为语言是通信的工具,可视语言的理论试图找到解释这种通信过程的理论基石。可视语言中的形式方法则为创建这样的可视语言的理论提供必要的数学工具。需要强调的是,形式化方法本身并不是理论,并且也还未得到充分的发展。十年来,可视语言中的形式方法的进展主要集中在三个方面:可视语言的形式模型,可视程序设计语言的语义和可视推理(Visual reasoning)。

正文语言的形式模型,诸如正规文法,上下文无关文法、上下文有关文法等,已比较成熟。在六十年代后期,人们就已开始从事图象语言(Pictorial language)的形式模型方面的工作,并提出了一些图象文法(Picture grammar),由于这些文法主要应用于模式识别,所以这方面的工作,K. S. Fu 在他的《句法模式识别》一书中作了较完整的概括。

80年代后期,由于受可视接口、可视程序设计和可视化最新进展的刺激,人们对可视语言的形式方法又有了新的兴趣,但重点已转移到人造可视语言的形式规范等方面。这种人造可视语言的一个子类便是 ICON 语言,在 ICON 语言中,每一条可视语句均由 ICON 的空间排列组成[3]。ICON 语言的形式规范问题得到了较充分的研究,人们已提出了用于 ICON 多种语言的文法,如关系文法(Relational grammar),图象布局文法(Picture-layout grammar),图文法(graph grammar),位置文法(positional grammar)等。每一次 VL 会议论文集均有大量文章报道。另外,JVLC(Vol. 2,NO. 4,1991)还出版了关于可视语言理论的专辑。

另一个受到广泛重视的可视语言是注释(annotation)。注释是交流思想的非常重要的可视语言,但目前尚无注释的形式模型。除此之外,用于复杂的二维或三维图象的可视语言也需进行深入研究。

可视语言的语义是一个更为棘手的问题,因为可视符号通常具有多重含义,可视符号的结合也会导致多种解释[4]。在可视程序设计语言中,这种二义性可通过赋予可视符号一个唯一的含义来解决。也就是在这一领域,可视语言的语义问题有很大进展。其中最著名的是关于可视程序设计语言的说明性方法(declarative approach),Ambler 及其合作者利用传统程序设计语言的理论为多种可视程序设计语言提供语义,他们的工作在最近的 VL 会议录中均有报道。另外,JVLC 还出版了关于说明性可视语言的专辑(Vol5,No. 1,1994)。另一个趋向是利用逻辑程序设计作为某些可视程序设计语言的语义基础。面向对象的程序设计,特别是用于多媒体扩展的可视程序设计语言,将会是引向进步的又一途径。

可视语言的形式方法的第三个方面是可视推理。可视推理是推理的过程和依据可视表示的线索进行的推理。可视推理已广泛应用于人与人之间的通信,例如,老师在黑板上画一个简图,尽管这个简图是不完全和不精确的,学生仍能够进行推理以填充相应的细节,并获得所表现概念的理解。这种简图理解(diagram understanding)依赖于可视推理以使概念得到交流。

在可视推理中,人们假设存在一种可能能够也可能不能够很好定义的基本的可视语言。从这种可视语言给定可视线索后,我们便可检查其可视推理过程。在这种一般的前后关系中,Del Bimbo 和 Jungert(VL'93)以及 Dillon's(JVLC, Vol. 5, No. 1, 1994)提出了空间推理(spatial reasoning);Kutty et al(VL'93)提出了时态推理(temporal reasoning);Wang 和 Lee(JVLC, Vol. 4, No. 2, 1993)设计了一种用于可视推理的形式语义;Myers(VL'90)设计了一个示范演释系统(deductive by-demonstration System)等等。

VL 十年来在可视语言的形式模型,可视程序设计语言的语义,以及理解可视推理过程(包括空间推理和时态推理)方面的研究,已经朝可视语言理论的方向前进了一大步。当然在宣布可视语言具有足够的理论基础之前,还需作大量的工作。

## 4 可视程序设计进展

可视信息的使用鼓励每一个人都直接参与程序设计。利用可视信息的程序设计称为"可视程序设计",这里,"程序设计"并不一定意味着象在 C++, Prolog 等语言中的程序规范,而是包括如查询一个数据库和指定对软件的需求等许多活动。

可视程序设计的思想出现于70年代末或80年代初。尽管当时"可视程序设计"这一术语尚未获得普遍使用。早期可视程序设计系统的例子有应用于逻辑电路设计的系统、计算机游戏系统、办公信息处理语言如 QBE/OBE[5]及 FORMHL(Shu, VL'84)等。

从80年代中期开始,可视程序设计的研究转向直接开发通用的程序设计语言和系统。这一时期出现的通用系统有:基于过程模型而设计的 Pict[6],以及 BLOX(Glinert VL'86),Foms(Ambler VL'87),

Inter CONS(Smith VL'88)，Fabrik(Ludolph et al. VL'88)，以及 LIVE(Kojima et al. VL'89)等等，在这些原型系统的基础上，已开发出商用的通用程序设计系统，Serius 和 Prograph(Cox et al. VL'89)便是这样的例子。可视地支持程序设计活动的 CASE 工具也已出现。

然而，通用可视程序设计语言目前尚未被人们接受作为现有文本程序设计语言的替代物，其困难在于可视语句可能隐含的二义性。正因为如此，从80年代末和90年代初开始，开发局限于某一特定域的可视程序设计系统又变得活跃。这方面最典型的例子是超媒体(hypermedia)环境中的创造工具(authoring tool)。开发特定领域的可视程序设计系统的又一领域是所谓的接口建造器，即用于图形用户接口构造的用户接口管理系统(UIMS)。在 NeXT 机上运行的 Interface Builder 是这方面最有名的例子。

VL 十年来研究的经验表明，以计算机为中心的，基于文本的概念和技术不大适合于可视程序设计。事实上，过去的许多通用可视程序设计语言都是在已有的程序设计语言如 PASCAL、PROLOG 的基础上，增加一个可视外壳而得到的，这样的语言在实际程序设计环境中的应用不大成功。因此需要开发更先进的通用可视程序设计语言。基于例子的程序开发方案估计是最有前途的[1]。

HI-VISUAL[6]是一种基于例子的系统，这种基于例子的程序设计类型被称为"带例子的程序设计(PWE)"，LIVE，LEDA(Mima VL'93)，和 CALVIN(Bell 等，VL'93)是 PWE 的另一些例子。而象 Oak(Tonouchi 等，VL'92)，Druid(Singh 等，VL'90)，Mondrian (Lieberman VL'92, VL'93)，CBAD(Sasakura 等，VL'93)等基于例子的系统，具有从有限的例子推导出一般程序结构的能力，这样的系统称为"通过例子的程序设计(PBE)"，而在人工智能中，则称为"自动程序设计"。除此之外，一个"基于笔的接口"(Pen-based)可提供"通过手势进行交互"的一种可能实现，Hyperflow(Kimura VL'92)便是一个基于笔的可视程序设计系统的例子。

在 IEEE Computer 最近的一期关于可视程序设计的专辑中，报道了如下几个方面的新进展：将可视程序设计语言扩展，使其适于解决大的复杂的问题[10]；可进行配置的，以用于不同应用领域的可视程序设计环境[11]；支持可视程序环境自动生成的可视语言编译的编译[12]，以及用于创建面向领域的可视语言的工具[13]等。

可视程序设计的研究，还需要和心理学，艺术等不同领域进行合作，以及本领域中研究人员与工程师的相互合作，才能取得更大的进展。

本文较详细地综述了可视语言的三个主要方面：可视引喻，可视语言中的形式方法及可视程序设计十年来的研究进展及未来的一些研究方向，并对可视语言的研究历史进行了简要的回顾，试图给想了解这一领域研究状况的读者提供一点帮助。

"一图顶千字"，人类的思维是强烈地倾向于图形的。另据在美国加州的一家调查公司预测：1995年，可视开发工具的销售额将增长57%，而到1999年，预计将会有37.9亿美元的市场。因此，从事可视语言的研究，无论是在理论上，还是实践上都具有广阔的前景，是社会效益与经济效益并重的难得的研究领域。作者热切希望我国能有更多的有识之士来从事这项研究。

美国匹兹堡大学 S. K. Chang 教授为作者提供了大量的文献资料，在此表示衷心感谢！

## 参 考 文 献

[1] C. Batini 等，On Visual Representation for Database Query Systems，Proc. of the Interface to Real and Virtual Worlds Conference，Montpellier，France，March 22—26，1993

[2] S. K. Chang，Icon Semantics—A Formal Approach to Icon System Design，Int. J. of Pattern Recognition Artificial Intelligence，1，(3&4)，1987

[3] S. K. Chang(ed.)，Principles of Visual Programming Systems，Prentice—Hall，1990

[4] A. Cypher(ed.)，Watch What I Do：Programming by Demonstration，MIT Press，1993

[5] E. P. Glinert 等，Pict：An Interactive，Graphical Programming Environment，IEEE Computer，11(17)，1984

[6] E. P. Glinert(ed.)，Visual Programming Environments：Paradigms and Systems，IEEE Computer Society Press，Washington，DC，1990

[7] E. P. Glinert(ed.)，Visual Programming Environments：Applications and Issues，IEEE Computer Society Press，Washington，DC，1990

[8] M. Hirakawa，T. Ichikawa，An Iconic Programming System，HI-VISUAL，IEEE Trans. on Software Engineering，(10)16，1990

[9] M. N. Zloof，QBE/OBE：A Language for Office and Business Automation，IEEE Computer，Vol. 14，No. 5，1981

计算机科学 1996 Vol. 23 №. 2

# HPF——FORTRAN 最新版本

李新颖 李晓明

(哈尔滨工业大学计算机系 哈尔滨150001)

**摘 要** HPF(高性能 FORTRAN)公布于1993年,是当今世界上 FORTRAN 语言的最新版本。本文系统地介绍了 HPF 相对于以往 FORTRAN 版本(主要是 FORTRAN 90)的新特点,并举出了一个程序实例,供并行计算技术的研究者以及工程技术人员参考。

**关键字** FORTRAN 90,HPF,并行计算,数据划分。

FORTRAN 语言自从本世纪五十年代从 IBM 破土而出之后,一直备受工程技术人员的青睐,成千上万用 FORTRAN 语言编写的大型应用程序在全球各地被广泛使用,至今 FORTRAN 仍旧被认为是最受欢迎的数学计算语言。高性能 FORTRAN(High Performance FORTRAN,即 HPF)是这种古老语言的最新扩展集,除汇集了以往 FORTRAN 的全部特性之外,还具有能够实现数据在并行处理机上的划分、数据的并行操作等适用于现代各种计算机结构的语言特色,在国外被通过数据并行程序模型开发并行计算技术的研究人员所广泛采用。HPF 公布于1993年,目前尚不是一个被国际标准化组织承认的标准,当前 ANSI 和 ISO 认可的 FORTRAN 语言标准是 FORTRAN 90,它公布于1990年。

## 一、从 FORTRAN90到 HPF.

我们知道 FORTRAN90是在 FORTRAN77的基础上扩展了一些数据结构,如:数据指针、结构类型,和并行语句,如:数组操作 A=B*C,已经可以说是一种并行度很高的语言了。但分析起来它有三个弱点:

(1)在数据并行执行上,它对整个数组的操作可以用一条语句很方便地完成。如 sin(A),但对数组内个别元素的操作就无法很简便地并行化。如:对一向量中的个别元素求反,用 FORTRAN90就要写成:

```
DO I=1,N,2
  X[I]=-X[I]
```

这样的串行语句。

(2)它对数据的划分不做要求,这样在程序执行时就不能保证获得最高的性能。

(3)它没有为用其它语言写的程序提供接口。即使一段程序用其它语言可以更好地表达,也无法嵌入到 FORTRAN 程序中。

HPF 针对这些弱点进行了扩展,增加了并行机制(FORALL),数据划分的机制(DISTRIBUTE),以及为 HPF 和其它语言提供接口的方法等。这样上面(1)中的问题就可以用 HPF 写成并行语句:

```
FORALL I=1:N:2
  X[I]=-X[I]
```

可以看出 HPF 在 FORTRAN90基础上增加的这些内容事实上就是在并行计算技术的研究中也曾被广泛接受的以数据划分为特色的FORTRAN版

李新颖 硕士研究生,李晓明 教授,博士生导师。

[10]M. M. Burnett et al., Scaling Up Visual Programming Languages, IEEE Computer, Vol. 28, No. 3, 1995

[11]G. Karsai, A. Configurable Visual Programming Environment: A Tool for Domain-Specific Programming, same to [10]

[12]G. Costagliola et al., Automatic Generation of Visual Programming Environments, same to [10]

[13]A. Repenning, T. Sumner, Agentsheets: A Medium for Creating Domain-Oriented Visual Languages, same to [10]

[14]强军,王兵山,李舟军,可视语言:把思想变成程序的工具,计算机科学. Vol. 21, No. 4, 1994