计算机软件的用户界面及其开发环境*)

毛其晶 （中国科学院自动化研究所）

摘 要

本文分析了近几年计算机软件的用户界面开发环境的发展情况，研究了该领域所面临的主要问题及应当采取的策略。显然目前开发图形用户界面高费用的状况还未得到根本地改善，用户界面开发工具还未成为一种实用化的工具。笔者认为这主要是由于用户界面模型和用户界面描述语言这两个关键问题尚未解决。尽管我们在短期内还难于根本解决所面临的问题，但采用了合理的策略仍可能设计出实用化的用户界面开发工具，也很可能帮助我们找到解决问题的钥匙。

一、引 言

交互图形用户界面从一出现就很受用户的欢迎。高质量的用户界面目前已成为应用软件成功的重要因素，因而越来越多的软件采用交互图形用户界面。但图形界面的开发费用通常很高，研究表明[1]，一个具有高质量用户界面的软件的百分之五十到八十的代码是有关用户界面的，因而关于图形用户界面的开发环境是近年计算机软件一个十分活跃的研究领域。

自1982年ACM SIGGRAPH召开有关图形输入交互技术的专题研讨会并首次提出UIMS(用户界面管理系统）的概念至今，近十年过去了，专家学者们在该领域作了大量的工作。在软件市场上，在实验室中，我们可以发现几十，上百种旨在为界面设计者提供更好开发环境的系统。但已有的系统究竟在多大程度上帮助了界面设计者，图形用户界面开发高费用的状况是否得到了根本的改善呢？本文首先分析用户界面开发环境近年的发展情况，然后就该领域所面临的主要问题和应采取的策略进行讨论。本文中开发工具设计者，界面设计者和最终用户分别指用户界面开发环境的设计者，用户界面开发环境的用户和应用系统的用户。

*）国家自然科学基金资助课题

二、历史和现状

所谓用户界面开发环境的系统大致可分为两类：以程序库形式提供界面设计者使用的交互图形工具箱和以某种高级语言形式提供界面设计者使用的用户界面开发工具。下面分别分析它们各自的发展和现状。

1. 交互图形工具箱

交互图形工具箱的发展可分为三个阶段，即：七十年代末八十年代初发展起来的标准图形软件包，如：CORE，CGI，GKS，PHIGS等；八十年代中期发展起来的基于各种窗口管理系统的开发工具箱，如：MS Windows的SDK，X Window的X lib和X Toolkit等；八十年代后期开始发展的基于面向对象风范的用户界面成分库，如：InterView[2]等。大部分基于窗口管理系统的开发工具也在不同成度上采用了面向对象的技术。前两类软件目前已经比较成熟，基于面向对象的图形工具箱还正在发展。这些软件是当前界面设计者使用的主要开发环境。

工具箱为构造用户界面提供一整套基本的用户界面成分和操作功能，如图元的产生和操作，菜单的构造和处理，窗口的管理等等。使用工具箱开发用户界面，设计者不必为显示或输入设备的物理特性操心，可直接

从工具箱选择所需的界面成分搭成特定应用所需的用户界面。

比起早期从写驱动程序起开发交互图形系统，工具箱的使用的确减少了界面设计者的劳动，且开发出的应用系统具有较好的可移植性。但由于工具箱是以库函数形式提供给用户的，因而其使用对象只能是训练有素的程序员，即使对这些程序员来说，用工具箱开发图形界面的费用和效率也还远不能令人满意。

2. 用户界面开发工具

用户界面开发工具的任务在于帮助界面设计者进行交互图形界面的说明，设计，产生快速原型，实现，运行，评价，修改和维护。有别于图形工具箱，界面开发工具通常以某种语言的形式提供用户使用。按照理想的情况，开发工具应当能根据界面设计者对所需界面功能的描述自动生成用户界面，即实现所谓自动程序设计。由于自动程序设计的基础研究有许多基本问题尚待解决，在短期内还不可能实现用户界面程序设计的全部自动化。尽管如此，由于具有部分的程序设计自动化，直观易学的设计环境等特征，用户界面开发工具仍能为改善用户界面开发环境作出很大贡献。

用户界面开发工具成为一个独立的研究领域始于UIMS（用户界面管理系统）概念的提出。UIMS的基本思想是，软件的用户界面和应用部分应完全分开，从而可提供专门对用户界面进行管理的管理系统（类似于数据库管理系统的思想）。1985年在德国Seeheim召开的有关UIMS的专题研讨会上提出了Seeheim模型，该模型描述了UIMS运行系统的逻辑结构，是第一个用户界面的抽象模型，因而对UIMS的研究有着重要的意义。

已实现的UIMS系统的功能主要集中于用户界面的屏幕设计和对话控制设计，如：Domain/Dialogue[3]，Menulay[4]，Rapid/USE[5]，Syngraph[6]，University of Alberta UIMS[7]等等，其中一些系统已成为产品进入市场。UIMS系统所采用的屏幕设计语言可分为甚高级文字语言和可视语言(Visual Language)两种。甚高级文字语言通过提高语言的层次来减少界面设计者的工作量；可视语言除了甚高级语言所具有的优点外，还具有直观易学等优点。UIMS系统所采用的对话控制设计语言主要有：状态转换图，上下文无关文法和事件语言三种[8]。近几年，越来越多的研究人员认识到与应用有关的图形处理在用户界面开发中所占重要地位，并致力于有关图形自动生成问题的研究，如：APT[9]，Garnet[10]，Micro-UIDT[11]等系统都体现了这方面的工作。

研究用户界面开发工具，不仅涉及计算机图形和用户界面技术方面的知识，还需要许多软件的其他领域的知识，如：人工智能，软件工程等。有关用户界面的新技术以及软件其他领域的新技术不断对UIMS的发展产生极大的影响和冲击。其结果一方面使UIMS自身的面貌不断更新，另一方面也使不少人对UIMS研究最初提出的思想产生疑问。有些系统尽管仍被其设计者称之为UIMS，而实际上此时UIMS包含了比过去更广泛的含义。在许多场合下，UIMS成为用户界面开发工具或环境的代名词。

用户界面开发工具目前还未成为多数界面设计者使用的一种工具，设计者一方面抱怨使用工具箱开发用户界面效率太低，另一方面却继续使用它。其原因在于开发工具目前尚不成熟，能够提供的功能十分有限，因而设计者使用这种工具必然受到许多限制。另外用户界面技术自身发展十分迅速，开发工具设计者所追求的是一个移动着的目标[12]，因而开发工具只可能支持在设计该工具时流行的用户界面。这一切都使得想要获得高质量用户界面的设计者眼下还必须使用工具箱。

显然，目前开发图形用户界面高费用的状况还未得到根本的改善，交互图形工具箱

由用户界面开发工具都还有待于进一步发展，特别是要使界面开发工具实用化，真正发挥它所应具有的作用，还要有一个过程。

三、新技术的冲击

前面提到新技术给用户界面开发工具的研究带来的冲击，这种冲击主要来自以下三个方面。

1. 直接操纵式界面

美国Apple公司的Macintosh系列机的操作系统采用直接操纵式的用户界面所获得的巨大成功，使越来越多的系统采用这种形式的界面，并已成为界面开发工具近年的发展趋势。直接操纵式界面比传统的界面与系统的应用部分之间具有更紧密的联系，这主要表现在其丰富的语义反馈。但按照UIMS的将软件的界面与应用部分完全分开的假设，UIMS是否能适用于直接操纵式界面呢？一些学者认为应该抛弃完全分开的假设，另一部分人则认为应该坚持完全分开的思想[12]。但从设计的角度看，用户界面的设计不能与应用完全分开而是要强调系统设计的观点已为多数学者公认[13]。显然，开发工具设计者面临新的研究课题，人们一方面在研究适合于直接操纵式界面的界面模型，同时也开始考虑用户界面开发工具同软件工程工具合为一体的问题[14]。

2. 可视程序设计

可视程序设计是指某种以二维或多维方式来描述程序的系统[1]，其目的在于使程序设计更加容易。可视程序设计是近年在计算机软件领域兴起的一项热门研究课题，用户界面设计由于其自身固有的图形特性很自然地成为这项研究的一个重要应用对象。这项研究一方面促使更多的用户界面开发工具转向采用可视语言作为用户界面的描述语言，另一方面用于描述界面的可视语言能够更加形式化。面向用户界面描述的可视语言也是开发工具设计者面临的新课题。

3. 面向对象的程序设计

面向对象的程序设计的最主要优点在于可编制具有复用性的程序，从而可大大减少软件设计中的重复性劳动。图形用户界面是当前应用面向对象技术最为广泛的一个领域，具有直接操纵式界面的系统大多是采用面向对象技术实现的。由于基于面向对象的用户界面与传统的用户界面的抽象运行机制不同，因而基于面向对象的用户界面模型是开发工具设计者面临的又一新课题。

除了来自上述三方面的冲击外，人工智能技术也已经并将继续对用户界面开发工具的研究产生很大的影响。如：乔治华盛顿大学研制开发的UIDE[16]采用了知识库的技术，该系统可根据设计者对用户界面功能的描述变换产生出若干在功能上等价的用户界面供设计者选择。Peridot[17]则使用了自动程序设计中基于实例的程序设计方法，系统根据设计者给出的有关用户界面行为的演示自动推演其一般行为并生成相应的界面程序。

显然，基于面向对象风范的直接操纵式界面将逐步成为用户界面的主要形式，因而相应的图形工具箱，用户界面开发工具也将是界面开发环境今后若干年内研究的主要对象。对于用户界面开发工具，可视程序设计，人工智能技术的应用，界面开发工具与软件工程工具的一体化则是今后发展的主要方向。

四、面临的主要问题

用户界面开发工具目前还存在许多理论和技术问题，其中用户界面模型和用户界面描述语言是设计实用的用户界面开发工具亟待解决的两个最关键的问题。此外，对图形约束关系的描述和开发工具的可扩充性也是当前影响开发工具实用性的主要问题。尽管其他形式的用户界面还将长期存在，但基于面向对象的直接操纵式界面无疑是今后几年内用户界面及其开发环境领域的主要研究对象，本文以这种界面为讨论的重点。

1. 用户界面模型

用户界面模型是指用户界面运行系统的抽象模型，是用户界面开发工具的基础，将

影响用工具生成的用户界面的功能和响应速度。从事用户界面开发工具研究的学者们为界面模型的建立花费了相当多的精力，近几年的工作主要集中在基于面向对象的模型的研究，但至今尚未有一种令人满意的模型。建立模型的难点在于：(1) 用户界面软件十分复杂，很难找到一种能满足用户界面的各种需要和各种用户界面的需要的模型；(2) 由于用户界面技术自身的发展十分迅速，已建立的模型常常满足不了新的用户界面技术的要求；(3) 基于所建立模型的用户界面程序必须满足一定的速度要求才具有实用价值，因此用户界面模型必须具有尽可能好的健壮性，可扩充性，并满足一定的响应速度。

基于传统程序设计的用户界面模型是通过用户界面开发工具实现的，而基于面向对象的用户界面模型则通过工具箱一级的用户界面成分库实现。尽管有些界面开发工具允许界面设计者通过工具对成分库进行扩充，但基本成分库必须靠手工建立。实践表明，这个靠手工建立的基本成分库一定要在质量上足够好，在功能上足够丰富，才能满足开发工具的需要，这意味着建立这个库的费用是相当高的。因此，对于基于面向对象的用户界面开发工具来说，建立一个好的用户界面模型显得特别重要。

基于面向对象的用户界面模型主要有以下三种：

(1) **基于交互技术的模型** 这种模型的基本思想是将交互技术作为用户界面的基本成分，IAMBUS[18]是这种模型的一个例子。该模型的基本成分是交互技术，由五个成分构成：触发(trigger)，提示(prompt)，反馈(feedback)，语义(semantics)和动态(dynamics)。交互技术被分为基本交互技术和复杂交互技术两个子类(subclass)，复杂交互技术可由基本交互技术和复杂交互技术构成，因而该模型具有较好的可扩充性。基于交互技术的模型的缺点是未包括有关图形的描述，因而是不完善的。

(2) **基于窗口的模型** 这种模型的基本思想来源于窗口管理系统，以窗口或类似于窗口的控制区作为用户界面的基本成分，因而一个用户界面成分涉及有关图形显示，人机对话以及同应用程序的接口的处理各个方面。各种基于窗口管理系统的图形工具箱多为这种模型，如X Toolkit是以widget[19]作为界面的基本成分。

基于窗口的模型的优点是概念清楚，使用方便。缺点是对界面设计者的限制较多；由于一个界面成分包括界面的输入输出各个方面，因而代码的重用性较差；且新的界面成分只能通过手工建立。

(3) **组合式模型** 由于用户界面软件十分复杂，靠单一形式的模型往往很难描述用户界面软件的各个方面。组合式模型的特点是将用户界面软件分解为几个部分（如输入和输出，或对话和显示等等）并分别建模组合使用。

Smalltalk MVC (model-view-controller)[20]属这种模型，该模型也是基于面向对象风范用户界面模型中影响最大的一个模型。其基本思想有些类似于Seeheim模型，即将用户界面分为应用(model)，图形(view)和控制器(controller)三部分。这个模型的缺点是，在实际中有些功能常常难于分清该属于哪一部分。

Garnet[21]系统类似于MVC模型，它用交互技术(interactor)作为控制器，用面向对象的图形系统实现图形，用普通Lisp语言实现应用。Garnet将交互技术概括为五类：用于菜单及按键(button)处理的Menu-Interactor，用于移动和缩放图形的Move-Grow-Interactor，用于建立新点的New-Point-interactor，用于描述方向的Angle-Interactor，用于描述文字处理的Text-Interactor，和用于跟踪用户行为轨迹的Trace-Interactor。

Garnet模型区别于MVC模型之处在于程

序员无需建立新的交互技术，而是通过为已有的基本交互技术设置不同的参数来满足不同用户的需要，从而避免了MVC模型的主要问题。但该模型的健壮性如何，是否便于设计者使用还有待于实践。

基于面向对象的用户界面成分库和用户界面目前都还未发展到成熟阶段，其原因除用户界面自身的复杂性和基于面向对象程序设计的历史尚短外，主要还是由于编制可重用软件是项费用相当高的工作，未经精心设计的重用软件是没办法重用的[22]。因此，要获得真正成熟的用户界面模型和相应的实用界面成分库至少还需几年的时间。

2. 用户界面描述语言

用户界面描述语言是指开发工具提供给设计者的设计语言，或者说是开发工具提供给设计者的界面，理想的用户界面描述语言应该符合人们一般的思维方式，即：类似于自然语言。但要构造这样一种语言是十分困难的，其原因一方面是由于找不到一组合适的，用于构造这种语言的基元。即：找不到一组基元既能满足构造各种用户界面的需要，又满足自然语言化的要求；另一方面是由于有关描述用户界面的语言是因有了计算机才诞生的，它从一开始就以面向计算机的方式存在，而无相应的自然语言。所以无现成可参考的自然语言，而要根据符合一般人思维方式的原则去借用或创造一种语言。目前用户界面开发工具使用的描述语言一部分仍是很算法化的，如前面提到的上下文无关文法，状态转换图和事件语言等等，另有一些则企图以符合人们思维方式的新面目出现，如Peridot[17]通过演示界面的实际行为来描述用户界面。

用户界面描述语言影响着用户界面开发工具的功能和性能，有关该语言的研究，特别是基于交互可视环境的用户界面描述语言的研究无疑将仍是本领域今后研究的重点。

3. 有关图形约束问题

所谓图形约束是指图形与图形之间及图形与应用之间必须维持的关系，图形约束问题是实现图形自动生成的难点。图形约束多种多样，如：基于某些规则的图形屏幕布置，基于应用变量的图形变化，受另一图形属性控制的图形等等都是图形约束的例子。事实上，只要涉及图形自动生成的问题就必然遇到图形约束问题，因而这也一直是研究图形自动生成问题中的重点。目前处理图形约束所采用的用户界面模型大致类似，可称为活动数据 (active data) 方法。具体方式为：定义某些图形属性值为动态，用某应用变量或另一图形的属性控制该值，这种约束关系靠统一的约束处理器 (constraint solver) 来维持。如：ThingLab and Animus[23], Micro-UIDT, Garnet, OPUS[24]等系统所采用的均为这种方法。现有描述约束关系的语言大致可分为两类：一类是面向界面模型的，另一类是面向问题的。面向界面模型的语言（如Micro-UIDT）太计算机化，而面向问题的语言（如Granet）目前的描述能力还很有限，有待于进一步研究。

4. 工具可扩充性

由于用户界面技术的迅速发展，使用户界面开发工具的可扩充性变得极其重要。目前为提高工具的可扩充性所采用的办法主要有两种，一种是使工具在结构上可扩充且扩充无任何限制，但扩充工作仍需由工具设计者来完成，因而工具更新通常较慢，且常常满足不了某些界面设计者的特殊需要；另一种允许界面设计者使用基本技术构造新技术（如使用基本交互技术构造复杂交互技术）。这种方法允许工具使用者对工具进行扩充，但通常所允许的扩充是十分有限的。在实际系统中，上述两种方法常常结合使用，即：有些界面成分允许界面设计者进行扩充，有些界面成分则需由工具设计者来扩充。

对于基于面向对象的用户界面开发工具，其扩充性常常表现为对新的对象类的扩充。由于可重用代码需要经过精心编制，因而大部分情况下新的对象类需手工编制。另

一种解决将新的对象类加入工具的办法是使用原工具，即构造生成界面开发工具的工具，这是一种手工与自动扩充相结合的办法，如Meta-UIDT[25]，可根据对新对象的简单描述自动对已有工具进行扩充。这种方法的好处在于允许设计者进行扩充，且新的对象类可以来源于任何地方，有利于在工具中集成来自各方面的智慧。但有关这种方法的研究目前还只是初步的，要达到实用还有许多工作要做。

五、策略

综上所述，要想在短期内实现完全自动化地开发高质量用户界面软件的目标是不可能的，在很长的一个时期内，能够提供给设计者的用户界面开发环境只能起一个助手的作用。即使如此，利用现有技术建立的用户界面开发环境仍可能成倍地提高开发用户界面的效率且不损失用户界面的质量，关键在于采用正确的策略，笔者认为以下几种技术的使用会有助于用户界面开发环境的研究获得成功。

1. 利用用户界面的一般知识

用户界面软件无法完全自动生成的一个重要原因在于很难找到一种语言唯一且准确地描述界面，利用用户界面的一般知识，根据设计者对界面的不完善描述自动生成界面原型，然后由设计者进行修改，这种方法可能成为解决该问题的一种途径。这一技术最近几年已越来越广泛地被采用，其中最常见的一种办法是利用缺省值，即：界面成分首先根据缺省值来建立，然后设计者根据需要进行修改。利用缺省值的方法通常只限于界面成分一级。另一种方法是根据设计者的说明自动构造若干在功能上等价的用户界面供设计者选择，如：UIDE[16]。还有一种方法是根据经验知识自动补充界面设计者未描述的部分，如：Garnet[10]可自动进行对话区（dialogue box）的屏幕布置，并允许设计者对自动生成的界面进行修改。类似的方法还有许多，概括起来讲就是自动生成加入的干预，比起那种要求设计者详细说明界面的所有细节的开发工具，这种系统的效率显然要高得多。

2. 较窄的范围

面向较窄的应用领域是另一种能够提高工具自动化成度的方法，如：Unidraw[28]只面向图形编辑器，APT[8]只面向关系信息系统等等。这类系统虽牺牲了一些通用性，但具有较高的自动化程度。由此而受到的启示是，假如能够将大部分的用户界面概括在不太多的几个领域中，且各自具有相应的高度自动化的工具，则用户界面自动生成的问题就可基本得到解决。

3. 面向不同的对象

当人们谈到工具时，比较容易想到的工具用户往往是非程序员。而事实上不仅非程序员需要工具，程序员甚至图形学专家同样需要工具的帮助，只是他们各自的需要不同罢了。构造满足上述各种用户需要的工具显然是很困难的，使工具的使用对象单一同样有助于提高工具的实用性，如：Hypercard[27]对非图形学专家来说是个极好的工具，但对图形学专家来说该系统具有的灵活性就太不够了。而大多数已有的用户界面开发工具则更适合于熟悉图形的程序员，对这些人来说自动化和手工可以并存，他们既需要自动化又需要灵活性。

上述的几种技术决非权宜之技，而很可能成为解决问题的钥匙，上一节谈到的几个问题无疑是我们所要研究的问题的重点，但合理的策略对我们获得成功也是非常重要的。（参考文献共27篇略）