可视化编程环境下人机界面的面向对象设计*

邵维忠, 刘 昕

(北京大学 计算机科学技术系,北京 100871)
E-mail: wzshao@pku.edu.cn; liuxin@cs.pku.edu.cn
http://www.pku.edu.cn

摘要: 可视化编程环境的出现给人机界面的开发带来了巨大的变化,使人机界面的实现可以通过可视化操作,以“所见即所得”的方式进行定制,并在环境的支持下生成程序代码.这种变化给人机界面的面向对象设计提出了新的问题:既然界面的实现主要不是靠手工编程,那么在实现之前还要不要进行设计?面向对象的设计阶段建立的类图还有什么用?旨在对此问题作出回答.首先论述这种条件下的人机界面开发仍然需要设计,但设计策略应当改进.然后给出针对可视化编程环境的人机界面 OOD(objected-oriented design)策略.该策略使设计工作大为简化,且更为有效、更适合基于可视化编程环境的人机界面开发.

关 键 词: 可视化编程环境;人机界面;面向对象;面向对象设计;类图
中图法分类号: TP391 文献标识码: A

人机界面的开发在整个系统的开发工作中占有很大的比例.在面向对象的设计(OOD)中,人机界面部分的设计是用面向对象的概念和表示法建立一个可实现的界面设计模型;其主要文档是一个描述界面成分的类图.在可视化编程环境广泛流行之前,各种 OOA&OOD 方法(例如 OMT[1],OOSE[2]以及对人机界面设计讨论最多的 Coad/Yourdon 方法[3])所给出的人机界面设计策略都基于这样一种假设——设计阶段所定义的对象类、它们的属性与操作以及它们之间的关系,都是要由程序员去编程实现的;在编程之前必须经过设计才能为编程提供依据.即人机界面也和系统的其他部分一样,需要经历分析、设计、实现、测试、维护等软件生命周期阶段.

然而,可视化编程环境的出现使这种原本看起来理所当然的观点遇到了困惑.在可视化编程环境中,应用系统开发者可以通过环境界面上的操作,以“所见即所得”的方式定制自己所需的人机界面.如此定义的界面对象将由环境所提供的工具自动地转换为应用系统的源代码.在这种条件下,人机界面的实现已经不是传统意义上的“编程”,不需要程序员逐行逐句地去编写每个对象类以及它的每个属性与操作.那么,通过 OOD 建立的人机界面类图还有什么用?类图中的类,既然不需要编程,又何必费时费力地在设计阶段去识别和定义它们?总之,可视化编程环境下的人机界面开发还要不要按 OOD 方法进行设计?如果需要,应该如何设计?

上述问题是一些长期从事软件开发的技术人员在使用 OO(object orientation)方法进行系统开发时提出来的.这些来自工程实践的问题表明,实现技术的进步对 OO 建模方法提出了新的研究课题[4].通过研究与实践,我们对此问题的回答是:在可视化编程环境下人机界面的开发仍然需要设计,但是设计策略应该改进,设计文档可以简化.下面首先讨论上述观点的依据,然后给出可视化编程环境下人机界面的 OOD 策略.

1 设计的必要性

在软件工程中,设计是软件实现之前的一个必要阶段.它的必要性主要体现在以下 3 个方面,实现手段的进

* 收稿日期: 2001-08-01; 修改日期: 2002-01-24
基金项目: 国家自然科学基金资助项目(60073015)
作者简介: 邵维忠(1946－),男,山东平度人,教授,博士生导师,主要研究领域为面向对象方法,软件工程环境,系统软件;刘昕(1976－),男,湖南张家界人,硕士生,主要研究领域为系统软件,面向对象方法,软件工程.

步并未使这些理由发生根本的动摇.

(1) 设计的主要目的是为实现提供依据,提供一份可实施的蓝图,即设计文档,然后让程序员根据设计文档去开发系统的源程序.数十年来编程技术在不断地进步,包括编程语言的改进、人机交互技术的提高、CASE 工具的出现等等.但是这一切只意味着编程效率的提高,在编程之前仍然需要设计.尽管可视化编程环境使系统实现方式从完全靠手工编码发展到可视化编程和部分程序的自动生成,但这也只是实现效率的提高,而不意味着在实现之前不需要设计.在进行可视化操作之前,仍需对以下问题有一个正确和高效的设计方案:

· 为了满足人机交互的需求,人机界面中要使用哪些界面对象?

· 交互过程中的各项输入和输出应由哪些界面对象完成?

· 如何通过界面对象类之间的各种关系体现人机交互命令的组织结构与层次?

· 如何通过界面对象和功能对象之间的消息实现它们之间的动态联系?

这些问题都需要通过设计来解决.如果不作设计就开始可视化开发,就很难得到一个整体效果良好、结构合理的人机界面,甚至可能隐藏一些逻辑上的错误.

(2) 设计的另一个目的是降低失败的风险.任何一个较大的软件,如果不经过精心设计就开始编程,那么一旦出了问题,将付出很大的代价.可视化编程环境使人机界面的实现变得很快捷,发现问题时重新开发一遍也不太费力,这似乎使失败的风险变得不那么严重了.但是有一点是改变不了的:不经过设计的界面开发,即使重新做一遍也难以保证有根本性的改进,仍可能产生许多新的错误.

(3) 与实现相比,设计是一种抽象层次较高的开发活动.按软件工程的常规做法,设计和实现是由不同层次和不同技术特长的人员分解担任的.这种分工使设计人员和实现人员分别承担不同的责任,关注不同层次的问题,有利于保证工程的质量,也使人材资源的使用趋于合理.

2 设计策略需要改进

在可视化编程环境下进行人机界面的开发,不但实现效率大为提高,设计阶段的工作也可以得到显著的改进.原因有二:

(1) 可视化编程环境一般都带有内容丰富的界面类库.类库中对大部分常用的界面对象都给出了类的源代码.充分地复用这些类是提高开发效率的关键.这要求 OOD 方法能够表示对这些类的复用.表示法应该足够简单,使设计文档中不必包含那些已经由类库中的类定义的内部细节;另一方面,对这些类和其他类之间关系的表达应该足够清晰.

(2) 界面对象的各种物理属性(如位置、形状、尺寸、颜色、风格等)是一种反映其外观形象的特征信息.由实现人员以“所见即所得”的方式直接定制这些特征,效果最好,效率也最高.相反,若由设计人员凭空构想,通过对象的属性把这些特征描述出来,然后再由实现者去实现,则很难达到理想的效果,而且效率很低.所以要有新的设计策略.这种策略应使设计人员只注意界面对象的逻辑特征,对它们的物理特征则统统忽略,留给实现人员去做决定.

3 一般条件下的人机界面 OOD 策略简介

人机界面的开发可基于不同的界面支持系统.按支持级别从低到高排列,有图形软件包、窗口系统、图形用户界面(GUI),支持级别越高,编程工作量越小[5];可视化编程环境提供了更高级别的支持,使界面部分的开发基本不需要手工编程.

一般条件下的人机界面OOD策略可以简单地概括为以下几点(由于不是本文讨论的重点,这里只作简单的介绍):

(1) 在设计之前首先要进行人机交互分析,目的是明确用户使用系统各项功能时的交互需求,从而得到一个组织合理的命令结构.

(2) 选定一种界面支持系统;了解它所提供的界面元素,包括各种元素的输入/输出功能、界面构造效果和实

现方式.

(3) 根据人机交互需求选择界面元素.决定由哪些界面元素完成人机交互中的输入和输出并体现命令的组织结构.

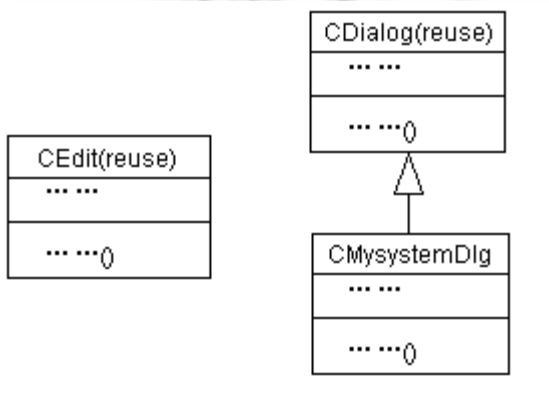
(4) 用面向对象的概念和表示法来表示所有的界面元素以及它们之间的关系,形成一个类图.其要点是:每个界面元素对应于系统中的一个对象,建立它们的对象类;用对象的属性和操作描述界面元素的各种静态特征及其各种操作;用整体-部分结构表示界面元素之间的物理构成关系以及由它们所处理的命令的逻辑层次;用一般-特殊结构表示较一般的界面类和较特殊的界面类之间的继承关系;用关联表示两类对象之间的静态联系;用消息表示界面对象之间以及它们和功能对象之间的信息传递.

4 基于可视化编程环境的人机界面 OOD 策略

在可视化编程环境及其类库支持下,采用改进的策略可以使设计工作大为简化.与以上讨论的一般条件下的设计策略相比,主要不同点是在其中的第(4)条,即类图的设计.下面我们详细讨论这种针对可视化编程环境的新策略,其中所用的例子是以 Visual C++及其类库 MFC 为背景的.

4.1 类的设立——首先想到复用

每当要建立类图中一个界面对象类时,应该首先想到使用环境(及其类库)所提供的可复用类.这些类通常能够满足应用系统的大部分人机交互需求.充分地复用这些类将大大地简化界面的设计和实现.



(a) Reuse directly (b) Reuse by inheritance

(a) 直接复用 (b) 通过继承复用

Fig.1 Reuse directly or reuse indirectly

图 1 直接复用或间接复用

类库提供的可复用类是针对一般应用的,应用系统在复用这些类时通常要在环境支持下进行定制.所谓"定制"有两种情况.一种是将可复用类的某些参数(例如属性的初始值)具体化.例如通过可视化操作确定编辑框控件 CEdit 的位置、宽度和高度,就只是将它的参数具体化,并未增加新的属性或操作.另一种情况是对可复用类的定义进行扩充.例如在 MFC 的对话框类 Cdialog 的基础上,通过填加若干控件,定制一个特殊的对话框类 CMysystemDlg.前一种情况是直接复用类库中的类;后一种情况是通过对可复用类的继承,定义了本系统中的一个新类.其表示法分别如图 1(a)和图 1(b)所示.

类库中提供的界面对象类通常都比较复杂,有许多属性和操作.另外在类库中往往有很多层被它继承的一般类.在应用系统的类图中全部表示这些信息既是一项沉重的负担,又将使类图变得很庞大,而且意义不大.所以我们给出一种简略的表示法——不填写被复用类的属性和操作,也不画出比它层次更高的一般类,只是注明这个类是被复用的.这种策略可以明显地简化 OOD 文档,并且使设计者将主要精力用于解决本系统的问题.对实现者而言,这种表示法已经能表明应该以哪个类为起点定制本系统所需的类.

4.2 属性表示——忽略物理特征,着重表示逻辑特征

通过继承可复用类而定义的新类是对可复用类的特化.通过属性体现的特化包括两种情况:一是对继承来的属性设置不同的初始值,二是在新类中增添新的属性.无论何种方式,设计阶段都不必关心描述界面对象物理特征的属性.诸如大小、形状、位置、颜色、边框、底纹、图案式样、三维效果等——这一切都由实现人员去自主处理,他们以可视化的方式定制界面对象的这些特征,效果会更好,效率也会更高.设计人员应该把主要精力用于定义那些描述界面对象逻辑特征的属性,特别是那些表现命令的组织结构、界面元素之间组成关系和关联的属性.例如在一个对话框中包含若干控件,应该用属性表示这个对话框含有哪些控件对象.

4.3 操作表示——显式地表示从高层类继承而且本系统需要关心的操作

可视化编程环境的界面类库通常是比较复杂的.在类树的各个层次上,每个类都定义了许多操作,都被应用系统定义的特殊类所继承.但是其中一些操作被继承之后并不真正被使用,另有许多操作是由编程环境在定制界面对象或者在支持界面运行时自动使用的,应用系统开发者不必关心.只有在手工产生的程序代码中需要调用的那些操作,才是设计人员和实现人员必须了解的.对于被复用的类采用第 4.1 节中所述的简略表示法可以避免类图中出现大量实现者不必关心的信息,但是真正需要关心的操作也看不到了.解决这一问题的补充策略是:在本系统定义的特殊类中显式地表示它是从类库中的类继承的,并且将在本系统手工编码中被调用的操作.其表示法如图 2 所示,在类符号的操作栏填写该操作的正式名称,并且做一个"√"标记,表明这个操作是通过继承得到的,不需要在本系统中实现,但是调用这个操作的其他对象的实现需要知道它的存在.这个例子中的 SetDlgItemText 函数是在比 CDialog 层次更高的窗口类 CWnd 中定义的,其功能是从窗口中一个指定的编辑框输出一段正文.程序员需要知道它的存在才能使用它.除了在类图中作显式的表示之外,同时在类的规格说明中指出该操作来自类库中的哪个类.

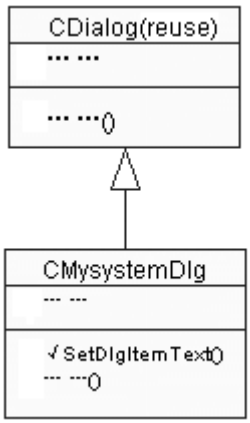


Fig.2 Representing the inherited operations explicitly

图 2 显式地表示继承来的操作

4.4 整体-部分结构——在特定条件下可以简化

通过整体-部分结构表现界面对象之间的组成关系和人机交互命令的层次关系,与第 3 节中介绍的一般策略没有太大不同,只是要注意以下几点:

(1) 区分对象的普通属性和它的部分对象.当一个界面对象带有内部组织结构时,可视化编程环境对不同的情况有不同的定义方式.有些组成部分(例如下拉菜单的选项,窗口的边框)被作为对象的一个普通属性;有些组成部分(例如对话框的按钮)则被作为一个部分对象.区分两种情况的依据是,环境类库有没有对这种组成部分给出相应的类定义.在 OOD 中,只对后一种情况建立整体-部分结构,前一种情况则只用普通属性表示,不要建立整体-部分结构.

(2) 在简单情况下可以隐式地表示部分对象.如果一个整体对象的部分对象具有下述简单性,则可以简化设计表示:

· 这种界面对象在本系统中总是依附于整体对象而存在,不会单独地创建对象实例.

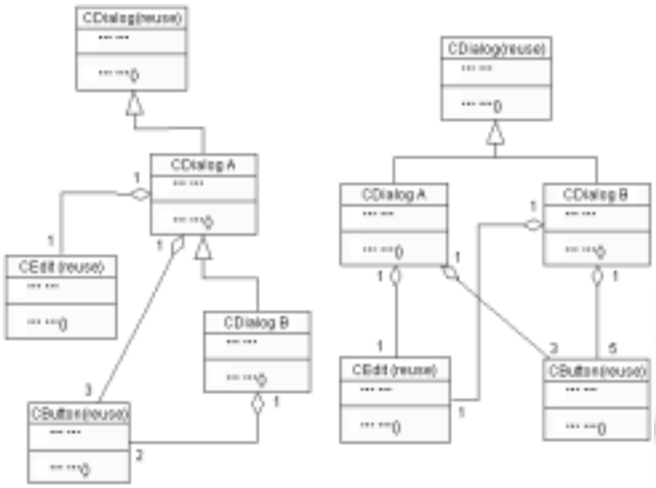
· 该对象不再包含级别更低的部分对象.

· 不需要由程序员通过手工编程实现该对象与其他对象之间的消息.

当部分对象全部符合上述条件时,整体-部分结构的表示法可以简化.即类图中不画出部分对象的类,只是在整体对象类中通过一个属性表示整体对象拥有这样一个部分对象.当类图很庞大时,采用这种策略可以使之简化.

4.5 一般-特殊结构——多从可复用类直接继承

如果一个应用系统中使用的多个界面对象有许多共同特征,按照通常的设计策略是运用一般-特殊结构,在一般类中定义共同拥有的属性和操作,特殊类继承一般类.例如,在一般条件下可以设计如图 3(a)所示的一般-特殊结构,使 A 继承可复用类,B 又继承 A,从而简化了这两个对话框类的设计和实现.但是在可视化编程环境下,采用如图 3(b)所示的一般-特殊结构可能更便于实现.因为在环境支持下,直接地基于可复用类来定制 B 类对话框是很方便的;相反,若按图 3(a)所示的结构,由基于自定义的 A 类对话框来实现 B,则可能缺乏环境支持,或者环境虽然有支持但操作比较复杂,需要有较高的技巧.



(a) Common strategy (b) Inherit from reusable class directly
(a) 普通策略 (b) 直接继承可复用类的策略

Fig.3 Different design strategies for generalization-specialization
图 3 一般-特殊结构的不同设计策略

4.6 消息的表示——忽略自动实现的消息

界面类库中的每个类都定义了许多操作,每个操作对应着一种消息.在这些类的基础上定制的类能够提供的操作(即能够处理的消息)也很多.但是其中有大量的消息(特别是处理界面对象常规操作的消息)是在可视化编程环境支持下自动实现的,不需要程序员通过手工编程去实现.因此设计类图时可以忽略对这些消息的表示.设计者需要关注的,是那些由程序员通过手工编程来实现的消息,包括以下几种情况:

· 由接收界面操作事件的界面对象向对该事件进行实际处理的功能对象发送的消息.

· 从要求在人机界面上进行输入或输出的对象向提供这种操作的界面对象发送的消息.

· 其他:如果所用的可视化编程环境自动化程度较低,那么凡是需要通过手工编程来实现的消息,都要在设计中加以表示.

5 结束语

可视化编程环境为人机界面的开发提供了强有力的支持,这种支持主要是针对实现阶段的.在实现之前仍然需要进行设计,但设计策略必须改进.本文给出了针对可视化编程环境的 OOD 策略,其要点是:(1) 强调软件复用,对被复用的界面对象类给出简单、明确的设计表示;(2) 清晰、准确地表示人机界面的逻辑结构以及界面对象与功能对象之间的关系;(3) 在设计中忽略界面对象的物理特征和其他能够在环境支持下自动解决的问题.这种策略使设计人员不必在 OOD 阶段做许多对可视化编程环境下的人机界面开发毫无意义的工作,而把主要精力放在真正应关注的问题上;使设计文档更为简练,但可以为实现提供更清晰、更有效的指导.

References:

[1] Jacobson, I. Object-Oriented Software Engineering, a Use Case Driven Approach. New York: Addison-Wesley Publishing Company, 1992.

[2] Rumbaugh, J., Blaha, M., Premerlani, W., *et al*. Object-Oriented Modeling and Design. Englewood Cliffs, NJ: Prentice-Hall, 1991.

[3] Coad, P., Yourdon, E. Object-Oriented Design. Englewood Cliffs, NJ: Prentice-Hall, 1991.

[4] Shao, Wei-zhong, Yang, Fu-qing. Object-Oriented Analysis. Beijing: Tsinghua University Press, 1998 (in Chinese).

[5] Dong, Shi-hai. Computer User Interface and Design Tools. Beijing: Science Press, 1994 (in Chinese).

附中文参考文献:

[4] 邵维忠,杨芙清.面向对象的系统分析.北京:清华大学出版社,1998.

[5] 董士海.计算机用户界面及其设计工具.北京:科学出版社,1994.

Object-Oriented Design of Human-Machine Interface in the Visual Programming Environment*

SHAO Wei-zhong, LIU Xin

(*Department of Computer Science and Technology, Beijing University, Beijing* 100871, *China*)

E-mail: wzshao@pku.edu.cn; liuxin@cs.pku.edu.cn

http://www.pku.edu.cn

Abstract: Visual programming environment has brought about great changes in the development of human-machine interface. It enables the developers to implement human-machine interface using visual operations by means of "what you see is what you get", and the program code can generate automatically by the environment. This change gives rise to some new problems to object-oriented design of human-machine interface. Since the implementation of the interface does not depend on manual programming, is it necessary to design before implementation? And what is the use of the class diagram built at the design stage? The answers to these questions can be found in this paper. It proves that the design is necessary yet in that environment, but its strategies should be improved. The OOD (object-oriented design) strategies for the development of human-machine interface are given, which makes the OOD work much simpler, more efficient and more suitable for development in the visual programming environment.

Key words: visual programming environment; human-machine interface; object-orientation; objected-oriented design; class diagram

* Received August 1, 2001; accepted January 24, 2002

Supported by the National Natural Science Foundation of China under Grant No.60073015