用户界面设计思想

“好用”的用户界面设计思想

□ 文／张亮

文章从系统模型和心理模型需要很好匹配的角度出发，通过引用大量软件产品的界面设计的例子，分析了为什么有些产品难以使用，并详细分析了该原则在编程语言的发展过程中的体现。最后，文章指出了如何使得系统模型能够和心理模型能够匹配的方法。

难用的垃圾桶

不知你是否留意过大街上或公共场所的垃圾桶，下面这幅照片显示的是北京某街道旁边的垃圾桶。

从该照片可以清楚地看到垃圾桶包括蓝色和黄色两部分，蓝色部分标明用来收集可回收物品，黄色部分用于不可回收物品。现在的问题是，作为普通市民，人们如何知道哪些物品属于可回收的。尝试问问你身边的人，恐怕连博士也未必能正确回答。

图1：标有使用方法但仍令人费解的垃圾桶

心理、实现和系统模型

以上关于垃圾桶产品难以使用的现象反应了一个很普遍的产品界面设计中的问题，即系统模型和心理模型的匹配问题。我们先来解释三个概念：心理模型，实现模型和系统模型，以及它们之间的关系。

Donald A. Norman在他的《The Design of Everyday Things》一书中首次提出了这三个概念及其关系。心理模型是存在于用户头脑中的关于一个产品应该具有的概念和行为的知识。这种知识可能来源于用户以前使用类似产品的经验，或者是用户根据使用该产品要达到的目标而对产品的概念和行为的一种期望。实现模型是产品的内部结构和工作原理，它存在于产品设计人员的头脑中。系统模型是指产品的最终外观以及产品呈现给用户后，用户通过观看或使用后而形成的关于产品如何使用和工作的知识。

从以上的定义不难看出，心理模型中的概念和行为是完全属于用户的问题

领域或任务领域的，而实现模型则位于技术解决方案领域。一般来说，这两者有很大区别，并且越是复杂的产品，差别越大。因为是位于问题或任务领域，心理模型是产品设计人员无法轻易改变的，而实现模型则依赖于当时的技术水平，在一定时期内也很难有大的变化，唯有系统模型具有极大的可塑性，是产品设计人员可以通过努力来改变的。可以认为系统模型总是分布于心理模型和实现模型这两者之间的某一点。系统模型越是接近心理模型，用户需要学习和记忆产品如何使用的地方就越少，这是因为实际的产品和用户期望的很接近，这样的产品就很容易使用。反之，如果系统模型接近实现模型，则用户需要把期望中的一些概念和行为映射到系统模型中表现出来的一些界面元素和执行操作上。这种映射在认知心理学上就表现为一种记忆负担，而正是记忆负担使人们觉得产品难以使用。

现在回到文章开头提到的垃圾桶难用的例子。在这个问题的心理模型中，有关的概念是将要扔掉的物品的名称，或最多包括一些朴素的关于物品的分类方法，例如食物类，包装类等等。而在实现模型中，因为要根据垃圾是否具有可收回性来做不同的处理，所以就有了"可回收"和"不可回收"两个概念。系统模型，也就是标有"可回收，不可回收"字样的垃圾桶的外观，几乎完全等同于实现模型而严重偏离了用户的心理模型。如果想要正确地使用这个垃圾桶，用户需要把日常生活中的各种物品是否可回收这一点完全背诵下来。对于那些记性不好的，则需要把这种对应关系记到一个小本子上，每次扔垃圾之前查阅一下。这就导致了一个易用性很差的产品。

系统模型和心理模型

由于系统模型和心理模型不匹配导致的产品易用性问题广泛存在于各种产品中。在这当中，软件产品的问题尤为突出。这一方面是由于软件产品固有的复杂性，另一方面，西方在文艺复兴以后，机械论逐渐成为一种世界观和设计思想体系，体现到计算机产品中，界面设计都是试图去让人适应计算机，而完全忽略了人的心理特点和需求。不过随着计算机越来越普及，计算机产品被越来越多的普通用户使用，"以人为本"的思想已经开始被越来越多的公司和产品设计人员所重视，即如何设计计算机的界面以使得其更好地适应人。

下面我们将重点分析一些软件产品中的这方面的问题。

个人电脑的文件系统

每当用户写完一篇文档（比如Word文档，Excel文档等）并要保存时，大多数软件总会弹出一个保存文件对话框来要求用户指定文件将要存到哪个目录下以及文件名。如果你对电脑的文件系统不了解，你就无从知道应该放到哪里。就算你放到了某个位置，一个月后你还能记得这个位置和文件名吗？对于大多数计算机操作系统来说，基于树状结构的文件系统完全是个实现模型中的概念，因为软件必须知道一个确定的位置才能进行保存文件的操作。而大多数普通用户的心理模型倾向于把计算机的存储设备看成一个简单的存储空间，只要能保存并且下一次运行软件时还能找到就可以了。按照这个思路，软件呈现给用户的界面中只应要求用户给文件起个名字就可以了，至于说到底把这个文件放到文件系统中的哪个目录，是由软件自身在实现时应该考虑的问题。例如，它可以把文件保存到其安装目录的某个子目录下，如果发现安装目录所在的磁盘没有足够空间时，它应该能够找到还有剩余空间的某个磁盘，在其上创建一个目录，然后把文件保存到那个新建的目录下。

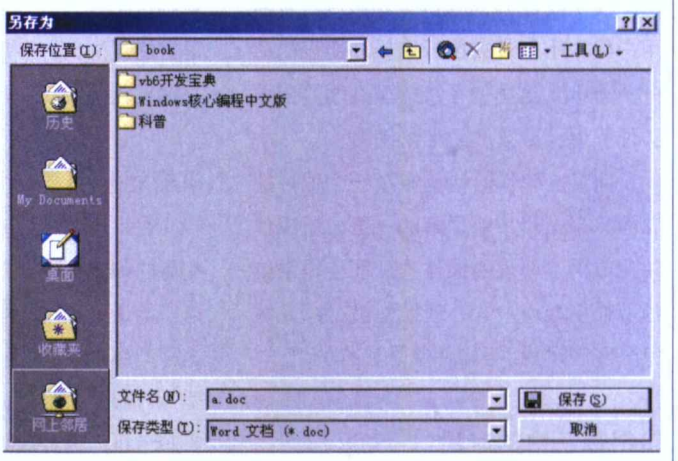

图2：Windows系统中的保存文件对话框

作为对比，同样是保存文件功能，诺基亚的所有基于S60操作系统的手机中的"记事本"软件就很好地把系统模型做得很接近心理模型。当你使用该软件编辑完一个文件时，按保存菜单，一切就完成了，没有了要求用户选择存储位置的对话框。更妙的是，用户连文件名都不用设定。当下次打开"记事本"软件时，所有你以前保存的文件都列了出来，每个文件占一行，显示的是文件开头的几个字，用户能够很清楚地知道该打开哪个文件。

Windows的注册表

在Windows中，操作系统的很多界面外观和行为是由注册表中的内容来控制的。如果对于计算机比较了解，知道如何修改注册表，并且知道具体是注册表中的哪个数值决定哪个外观或行为，你就可以对系统有更多的控制，从而把它定制成完全符合自身需求的环境。

例如，如果想要设置光标在屏幕上的闪烁速度，可以启动注册表编辑器（对应的命令是regedit），打开HKEY_CURRENT

_USER\Control Panel\Desktop子项，在该子项中有一个名为 CursorBlinkRate 的字符串值项，该值项的数值就是光标在屏幕上的闪烁速度。

在这里，注册表就是实现模型，用户想要修改的外观或行为是心理模型，而系统模型就是Windows操作系统。普通用户根本不知道去执行编辑注册表的命令 regedit。这里的系统模型几乎等同于实现模型，导致产品难以使用。

而某些软件恰恰是做了一个很好的界面，来把用户能理解的一些设置操作转化成对注册表的操作，从而使得系统模型很接近用户的心理模型。软件"超级兔子魔法设置"等软件就是这样的例子。

输出信息中的应用

大多数软件界面都由输入和输出两部分构成，在输出部分，系统模型和心理模型需要匹配的原则同样适用。软件设计中的"尽量使用用户或问题领域的语言"的原则就可以看作是该匹配原则的一个推论。具体来说，界面用语要使用用户能理解的词汇，包括所有反馈信息，例如提示和出错信息。这里举一个不恰当的出错信息的例子。某些网站的后台是用基于J2EE的技术实现的，在执行某些操作时，可能会出现程序异常，而此时如果没有捕获异常并进行相应处理，系统最终会将Java运行时的异常堆栈信息显示在浏览器中。这样的信息对于调试该应用的程序员来说也许是件好事，但估计大部分用户看了之后都会一头雾水，不知道到底发生了什么事情，不知道下一步该怎么办。

编程语言的匹配发展史

对于程序员来讲，可以将编程语言看做是某种软件产品，它的界面就是该语言的词法、语法和语义规则，使用该语言编程就可以看作是对该产品的使用。

机器语言到高级语言

假定程序员要求解的问题是求两个变量的和。先看机器语言，它是所有编程语言的最终实现模型，所以"机器语言"这种产品的实现模型完全等价于系统模型，因而是最难学习和使用的。程序员需要知道机器指令是如何由0和1这样的比特位构成的，以及每个比特位的意义。

汇编语言则把系统模型向程序员的心理模型推近了一步。使用汇编语言时，程序员只需要知道用mov 汇编指令把一个操作数放到某个寄存器中，以及用add 汇编指令把某个寄存器中的数值和某个操作数相加。至于这些mov 或add 汇编指令是如何对应到机器语言的，他不必知道，这是汇编程序要做的事情。不过很显然，到汇编语言这一步，程序员仍然需要了解"寄存器"和mov指令的概念，而这可不是问题领域中的概念。

再来看高级语言，比如C语言，系统模型和心理模型几乎完全一致了，程序员可以用下面的语句来解决两个数相加的问题：c=a+b;

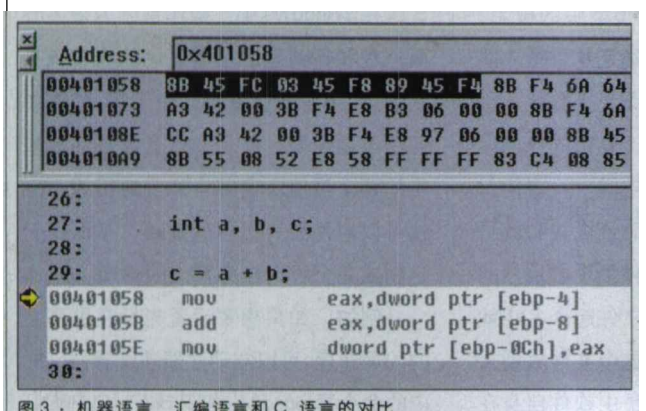

图3：机器语言、汇编语言和C 语言的对比

左面的图3 对比了三种语言的区别：其中黑底部分的16进制数是机器语言(Intel X86系列CPU的机器语言)，白底部分的文字是汇编语言(Intel X86系列CPU的在Windows平台上的汇编语言MASM)，"c=a+b;"这条语句是C语言。

随着要解决问题的复杂程度的提高，编程语言也在以下几个方向上不断地发展着，他们从不同的角度将语言的系统模型进一步推近心理模型：

面向过程到面向对象

在较为复杂的应用中，使用面向对象的分析方式来对问题领域进行需求分析显得更为自然和有效，也符合人的思维方式。面向对象的语言正是认识到了这一点，在语法和语义的层面上直接对面向对象的思维方式进行支持，使得系统模型更接近程序员的面向对象的思考方式。

从3GL 到4GL

以PowerBuilder，Visual Basic，Delphi等语言为代表的4GL 语言的最大的特征是将更多的底层操作系统的实现模型隐藏了起来，尤其是在开发一些具有丰富图形界面的应用程序时。下面比较分别用PowerBuilder语言和C语言(结合Win32 API)来实现在Windows操作系统中把一个窗口的底色改为黑色。

在PowerBuilder中，假设w_main代表一个窗口对象，那么只用下面一条语句就可以改变其背景色为黑色：

如果是用C 语言，并且假定该窗口是个对话框窗口(DialogBox)，则需要在该窗口的窗口处理函数中增加如下的处理（黑体文字部分)，

如果该窗口不是对话框窗口，而是一个普通的窗口，则处理方式又不相同。之所以会这么复杂，是因为在Windows操作系统中，当要绘制一个对话框窗口的背景时，操作系统会首先向该窗口发送WM_CTLCOLORDLG消息，消息的返回值就是将用来绘制背景的刷子对象的句柄。在C语言中，程序员必须对这个实现机制完全了解。而在PowerBuilder中，你可以认为那条简单的赋值语句帮你写出了类似C语言的那段代码。

```
w_main.BackColor = RGB(0,0,0)
```

```
LRESULT CALLBACK About (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
        switch (message)
        {
                ...
                case WM_CTLCOLORDLG:
                        {
                                HGDIOBJ Brush = GetStockObject(BLACK_BRUSH);
                                return (LONG)Brush;
                        }
                        break;
        }
    return FALSE;
}
```

通用语言到领域专用语言

应该说，这是编程语言发展的最终目标，它试图将语言设计得完全匹配程序员的心理模型。也许到那时，只要是对某个应用领域的业务熟悉的人都可以成为该领域的程序员了，因为编程语言中的概念和流程控制同实际工作中是一模一样的。

已经出现的这样的语言包括：SAP系统的ABAP语言，编译器软件开发领域的Lex 和 Yacc语言，数据库领域的SQL查询语言等。以Lex语言为例，它的应用领域是开发编程语言的词法分析器。假定你要设计的新语言的变量拼写形式是由一个或多个英文字母构成的，用Lex语言，这个需求可以描述为 [a-zA-Z]+。Lex的编译器会根据这个描述自动生成所需的词法分析代码。在这个例子中，[a-zA-Z]+ 就是问题领域的直接描述，Lex语言对它做了直接支持，即系统模型和心理模型完全匹配。

在产品设计中运用匹配原则

很显然，为了使得产品的系统模型能够很好地匹配心理模型，设计人员的首要任务就是要研究用户的心理模型到底是什么。近年来发展起来的以用户为中心的设计方法，就集中体现在它非常重视对于心理模型的研究工作。以下是一些常用的探究用户心理模型的技术，根据产品的不同特点，某些技术会比其他的更有效。

1．用户访谈：即和实际用户或潜在用户面对面交流，听取他们对产品的需求或期望；

2．用户现场观察：即在用户工作现场或将要使用产品的地方观察用户当前处理问题的方式，试图找出当前处理方式的问题或不足。这种方法对于新产品尤为重要，因为用户很难想象出一个他从来没有见过的产品的功能，而设计人员往往可以通过现场观察，分析出用户所要达到的目标，从而有可能在新产品中以一种新的方式来帮助用户达到这个目标；

3．竞争产品分析：这是因为市场上已经有了这类产品，用户就已经形成了对该类产品的一种期望，而通过分析这些产品，就能间接地找出这些期望。

4．分析用户对已有产品的反馈：这包括分析客户技术支持或客户投诉的历史数据，用户在新闻组上发表的对产品的评论等。从这些分析中往往可以看出有哪些用户的期望没有被得到满足，从而有助于形成一个更完整的心理模型。

在确定了用户的心理模型后，可以采用一些可用性原则，使得系统模型能更好地展现心理模型。这些原则包括反馈原则，可视性原则等。鉴于篇幅所限，此处不再展开，详细论述请参考Donald A．Norman的《The Design of Everyday Things》一书。

小结

对于用户来讲，产品的界面就是产品的全部，用户看不到也不用关心产品的内部结构和工作原理。界面设计的主要任务就是要把用户能理解的任务领域的概念和行为（心理模型）转换成产品内部的实现模型，从而向用户呈现一个尽可能接近其心理模型的产品（系统模型）。这样的产品能够减少用户的学习和记忆负担，从而达到提高易用性的目的。■

■ 责任编辑：欧阳璟 (ouyangjing@csdn.net)

参考书籍

《The Design of Everyday Things》
Donald A. Norman

《About Face 2.0: The Essentials of Interaction Design》
Alan Cooper

《Software for Use》
Larry L. Constantine Lucy A.D. Lockwood

作者简介

张亮，1997年毕业于中国科学院自动化所模式识别与人工智能专业，从事过汉语语音识别、B2B电子商务，网络通讯等领域的软件开发工作。