第 21 章：软件界面设计　　作者：希赛教育软考学院　来源：希赛网　2014年01月27日

输入/输出的识别与分类

第21章 软件界面设计

在应用系统中，界面的重要性越来越突出，甚至关系到系统能否成功实施。

界面对于开发人员而言，仅仅是部分，甚至被认为是皮毛之类的无关痛痒的部分，但对于用户而言，则是时时要面对的重要部分，甚至是全部。

界面设计的内容包括用户输入/输出界面样式、操作方式和界面间的转 也可以是开发工具编写的界面原型程序。界面设计的倾向应该是使用户感到操作软件是一 情；应该使用户感到软件是有礼貌的；尽量使用贴近用户环境的交互语言；最好花费一 界面原型程序，在编码之前就让用户充分测试，尤其是可用性测试。

例如，"保存"可能是应用软件中最常见的按钮，友好的界面应该是当保存出错时，弹出对话框"网络中断，保存失败！";但常常见到成功执行了保存，仍然弹出对话框"保存成功！",必须确定后才能继续，这就是不友好的界面。

21.1 输入/输出的识别与分类

在需求分析阶段，分析员已经标识出了关键的输入/输出，在设计阶段要进行详细的识别。

1.传统和面向对象的输入/输出

在传统的方法中，通过在数据流图中增加更多的细节数据流，从中识别输入/输出。

在面向对象方法中，进入和离开系统的消息就是要识别的输入/输出。用例图中，角色为用例提供输入，用例为角色提供输出；在交互图、设计类图中的方法，状态图中的转换，都可为识别输入/输出提供信息。

2.用户界面和系统界面

界面可分为用户界面和系统界面两种。

用户界面指系统中需要用户交互的输入/输出部分。这种界面非常直观，需要用户直接输入信息，会把输出信息展现到用户面前。

系统界面指很少需要人员干预的输入/输出部分。这种界面较为隐蔽，比如来自其他系统的电子信息、向其他系统发送消息或信息等。

这两类界面的设计需要不同的专业知识和技术，因此应该分开设计。

上一节　本书简介　下一节

第 21 章：软件界面设计　　作者：希赛教育软考学院　来源：希赛网　2014年01月27日

理解用户界面

对于最终用户来讲，用户界面就代表了系统本身，很多开发人员也认为设计用户界面就是设计

系统，应及早考虑界面设计。

1.用户界面的物理特征

物理特征一方面包括用户接触到的设备，如键盘、鼠标、触摸屏等，另一方面包括用户参考手册、输入/输出窗口等。

2.用户界面的感知特征

包括用户看到、听到、触摸到的所有东西，如屏幕上显示的窗口、数据、文字，语音识别指令，借助于鼠标对屏幕上的对象进行"触摸".

3.用户界面的概念特征

要想使用系统，用户必须了解很多系统的细节，必须对系统的运行方式有清楚的认识，对业务处理的过程也要清楚。也就是说要对在屏幕上展现的系统界面上完成自己的业务的过程、顺序了解得很清楚。

4.以用户为中心的设计技术

设计中更多地为用户考虑，努力提高系统的可用性，已经成为软件开发人员的共识。为此我们应该做到以下几点：

1）需要及早地关注用户及其工作

面向对象的方法相比传统的结构化方法，由于面向对象系统更加具有交互性，因此更加关注用户和他们的工作，用户作为系统的角色，应始终受到关注。

最好能尽早地就界面与用户进行沟通，在这方面，快速界面原型工具就是必不可少的。通过建立仿真式的界面原型，使得用户可以直观地体验未来的系统操作过程，从而更加有的放矢地提出意见。

2）反复评价系统设计以确保其可用性

由于系统最终用户的差异，要保证对于每一个用户，系统都有很好的可用性并不容易。需要我们收集、整理用户的各种素质，抽象出几个典型的角色，系统至少要让这些典型的角色感到系统良好的可用性。

3）使用迭代开发方法

通过几次迭代，每次迭代都以用户为中心，系统会越来越符合最终用户的要求。

4）人机界面研究领域

正是由于界面变得越来越重要，许多有志之士都投入到此领域，展开深入的研究。人机界面被上升到理论的高度。其中施乐（Xerox）公司做出了很大的贡献，他们建立了Xerox Palo Alto 研究中心（Xerox PARC），专门研究涉及影响人对机器操作的问题。研究中心开始运作后，可谓硕果累累，著名的面向对象语言Smalltalk,就出自该中心Alan Kay之手。现在著名的苹果（Apple）公司也从为该中心进行的开发中获益匪浅。该中心提出的很多先进设计理念和开发技术正日益集成到众多商业系统的开发方法中。

上一节　本书简介　下一节

界面设计指导原则

为规范和引导界面设计工作，需要制定相应的原则予以保障。

1.可视性和可供性

Donald Norman提供了这两项原则。

可视性 规定所有的控件必须是可见的，并且控件在用户动作之后应提供及时的反馈以示响应。比如按钮按下后形状变化或发出声响。

可供性 也就是规定所有控件的外观都应该体现和反映控件所实现的功能，比如滚动条提供内容滚动功能。

如果设计的界面能满足上述原则，那么这个界面将会具有良好的直观性和交互性，就是好用的。

2.美观耐看

仅仅具有直观和交互对于越来越成熟的用户来说是不够的，因为用户要在相当长的一段时间内频繁地面对界面。赏心悦目的色彩和错落有致的布局会使用户感到愉悦，创造性的操作过程或展现方式会使用户感到惊喜，这样的界面是吸引用户的。但要达到这样的目的却是不容易的，没有现成的方法可供采用，但只要在设计界面时尽可能地考虑这方面的因素，就会得到用户一定的正面响应；反之，抱着"酒香不怕巷子深"的老观念一味强调界面的功能性而忽视用户的审美取向的设计，必然是失败的。

3.调节界面

程序一般要提供大量的功能，但并非在任何时间、任何地方对全部用户都需要。通常是只需要一个小的子集，随着用户承担角色的变化，这个子集还有可能改变。我们设计时需要考虑这种个性化的因素，突出其关心的部分，把所有其他的放到次要的位置，使其离开正常的视野，经过这样的调节，界面将达到简化。

4.8条黄金规则

Ben Shneiderman总结出了8条基本设计原则。

1）尽量保持一致性

设计外观和功能的一致性是界面设计的重要目标之一。信息在窗体上的组织方式、菜单项的名称及其排列、图标的大小和形状，以及任务的执行次序、同样错误的提示信息等都应该是贯穿始终的。人有习惯性，设计应该向好的习惯靠拢。

2）为熟练用户提供快捷键

长期使用某个应用系统的用户愿意花些时间学会快捷键的使用，而且一旦熟练掌握了快捷键后，就对菜单式的操作和大量的对话框失去了耐心。所以设计者应该提供快捷键，最好提供自定义快捷键的功能。

3）提供反馈信息

对用户所做的每一个动作，系统都要提供某种类型的反馈信息，使用户知道相应动作是否已被确认。没有任何反馈信息会让用户无所适从。

4）设计完整的对话过程

系统的每一次对话都应该有明确的次序：开始，中间处理过程，结束。如果用户想"查一下账户余额",那么对话过程将以单击"余额查询"按钮开始，接下来是进入余额查询界面显示余额，然后单击"关闭"按钮结束。

5）提供简单的错误处理机制

一旦系统发现错误，错误信息应该特别说明出了什么错并且解释要如何改正。错误信息不应该是指责性的。系统还应简化错误处理。比如，用户输入一个无效的账号，系统应提示用户并把光标停留在账号上以供编辑，用户不必完全重新输入。同样的错误，如果系统只提示"输入的用户信息无效，请重试",之后，清空原先所有输入的数据以便重输，那么用户仍然不知道具体什么地方出了错，而且还要全部重新输入一遍。这样会大大挫伤用户使用系统的积极性。

6）允许撤销动作

试探是用户学习使用系统的一种方法，应该让用户感觉他们可以毫不费力地撤销相应的动作。这也是防止出错的一种方法。设计者应该在所有对话框中都包含取消按钮，允许用户在任一步骤都可以回退。

7）提供控制的内部轨迹

有经验的用户希望有控制系统的感觉，系统响应用户命令，而不该使用户感觉被迫做某事或者感觉正在被系统控制。系统应该让用户觉得是由用户自己在做决定。设计者可以通过提示信息使用户产生这种感觉，如"正在执行查询命令"等。

8）减轻短期记忆负担

短期记忆是人类本身的限制，人在同一时间只能记忆7条信息。因此，界面设计者不能假设用户能够记住在人机交互过程中的所有内容。

遵守上述8条黄金规则，就能保证用户界面的高效和可用。

5.专业性

原则上，软件的交互界面设计应由专业的交互设计人员承担，而不是由软件工程师来设计。因为大多数的程序员更多考虑的是功能的实现，实际上他们对交互界面该如何设计是缺乏经验的。界面设计需要大量的工作，比如，了解用户的操作习惯和操作现状，了解用户对于计算机的熟悉程度，了解哪些是用户频繁操作的功能，还需要反复的迭代设计，必须由专门的人员来全心投入。

6.特定性

一个软件会有很多的用户，俗话说"众口难调",我们该如何用最少的时间满足绝大多数用户的要求呢？引入有代表性的"角色"是一个好方法。一般的软件的使用用户无非是企业的高层领导、管理中层和一线作业人员这几类，可以针对这3个层次的用户分别设置"角色",比如"局长"、"科长"和"车工".但仅仅到这一步还是不行的，还需要对这些抽象出来的"角色"进行具体化，比如"王海局长"、"吕明科长"、"文秘小王".接着，要对这些角色赋予一定的细节。比如对"文秘小王",假设"她"是个女人而不是男人，主要是因为女秘书占大多数。这些定义好的具有具体细节的角色，要求全体开发人员都要熟悉，要改变开发人员经常挂在嘴边的"用户",而是用"王海局长"、"吕明科长"、"文秘小王"等来代替。我们把各种对于用户的讨论，具体化为对这些角色的讨论，将能有效地消除"用户"在开发团队中的模糊感。我们把这些"角色"放到软件可能的运行场景中，来观察和分析应如何设计交互界面。采用这样的方法，我们可以很快地获得最接近实际的好的设计。

上一节　本书简介　下一节

UML概述

第22章 UML分析与设计

UML（Unified Modeling Language,统一建模语言）已经日益成为建模标准，应用越来越广泛，现在已经成为了软件分析与设计建模的标准。作为软件分析与设计人员必备的技能之一，UML分析与设计现已纳入了软件设计师级考试大纲之列，并在2004年上半年的下午考试中出现。

22.1 UML概述

在20世纪的80~90年代，面向对象的分析与设计（OOA&D）方法获得了长足的发展，而且相关的研究也十分活跃，涌现了大量的方法学，据不完全统计，最多的时候高达50多种。其中最有代表性的当数Booch（Grady Booch）方法、OMT（Jim Rumbaugh）、OOSE（Ivar Jacobson）3种，而UML正是在这3位大师联手之下，共同打造而成的，现已成为了标准的建模语言。

上一节 本书简介 下一节

第 22 章：UML分析与设计 作者：希赛教育软考学院 来源：希赛网 2014年01月27日

UML是什么

UML（Unified Modeling Language,统一建模语言）是用于系统的可视化建模语言，尽管它常与建模OO软件系统相关联，但由于其内建了大量扩展机制，还可以应用于更多的领域中，例如工作流程、业务领域等。

UML是一种语言：UML在软件领域中的地位与价值就像"1、2、3、+、-、..."等符号在数学领域中的地位一样。它为软件开发人员提供了一种用于交流的词汇表，是一种用于软件蓝图的标准语言。

UML是一种可视化语言：UML只是一组图形符号，它的每个符号都有明确语义，是一种直观、可视化的语言。

UML是一种可用于详细描述的语言：UML所建的模型是精确的、无歧义和完整的，因此适合于对所有重要的分析、设计和实现决策进行详细描述。

UML是一种构造语言：UML虽然不是一种可视化的编程语言，但其与各种编程语言直接相连，而且有较好的映射关系，这种映射允许进行正向工程、逆向工程。

UML是一种文档化语言：它适合于建立系统体系结构及其所有的细节文档。

上一节 本书简介 下一节

第 22 章：UML分析与设计 作者：希赛教育软考学院 来源：希赛网 2014年01月27日

UML的发展历史

面向对象建模语言最早出现于20世纪70年代中期，而在80年代末开始进入快速发展阶段。截止到1994年，就从不到10种发展到50多种。由于每个语言、方法的创造者都极力推崇自己的成果，于是爆发了"面向对象技术的方法大战",也从此流传着一句戏言："方法学家和恐怖分子的差别在于，方法学家不能谈判。"

而在1994年之后，各种方法论逐渐拉开了差距，以Grady Booch提出的Booch方法和Jim Rumbaugh提出的OMT（Object Modeling Technique,对象建模技术）成为了可视化建模语言的市场老大。而Ivar Jacobson的Objectory方法则成为最强有力的方法。

Booch是面向对象方法最早的倡导者之一。他在1984年的《Ada软件工程》（Software Engineering with Ada）一书中就描述了面向对象软件开发的基本问题。1991年，他在《面向对象的设计》（Object-Oriented Design）一书中，将以前针对Ada的工作扩展到整个面向对象设计领域。他对继承和类的阐述特别值得借鉴。Booch1993比较适合于系统的设计和构造。

Rumbaugh等人提出了面向对象的建模技术（OMT），采用了面向对象的概念并引入各种独立于程序设计语言的表示符号。这种方法用对象模型、动态模型、功能模型和用例模型共同完成对整个系统的建模，所定义的概念和符号可用于软件开发的分析、设计和实现的全过程，软件开发人员不必在开发过程的不同阶段进行概念和符号的转换。OMT-2特别适用于分析和描述以数据为中心的信息系统。

Jacobson于1994年提出了面向对象软件工程（OOSE）的方法。其最大特点是面向用例，并在用例的描述中引入了外部角色的概念。用例的概念贯穿于整个开发过程（包括对系统的测试和验证），是精确描述需求的重要武器。目前在学术界和工业界已普遍接受用例的概念，并认为其是面向对象技术走向第二代的标志。OOSE比较适合支持商务工程和需求分析。

1994年10月，Grady Booch和Jim Rumbaugh开始致力于这项工作。他们首先将Booch93和OMT-2统一起来，并于1995年10月发布了第一个公开版本，称为标准方法UM0.8（Unified Method）。1995年秋，OOSE的创始人Ivar Jacobson 加盟到这项工作中。经过Booch、Rumbaugh和Jacobson 3人的共同努力，于1996年6月和10月分别发布了两个新的版本（UML 0.9和UML 0.91），并将UM重新命名为UML.

1996年，UML被OMG提议为OO可视化建模语言的推荐标准，UML被提交。1997年，OMG采纳了UML,一个开放的OO可视化建模语言工业标准诞生了。现在UML已经经历了1.1、1.2、1.4、1.5、2.0、2.1这6个版本的演变，已经发布了最新的2.2版标准。如图22-1所示。

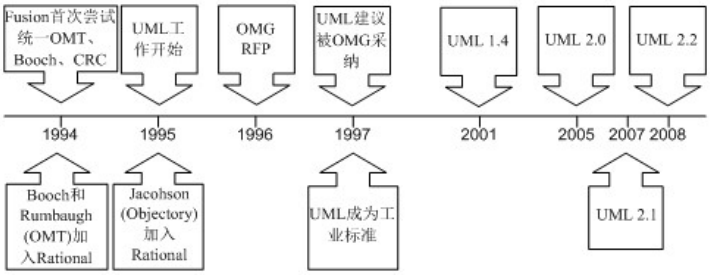

图22-1 UML发展历程示意图