HCI 复习

什么是人机交互

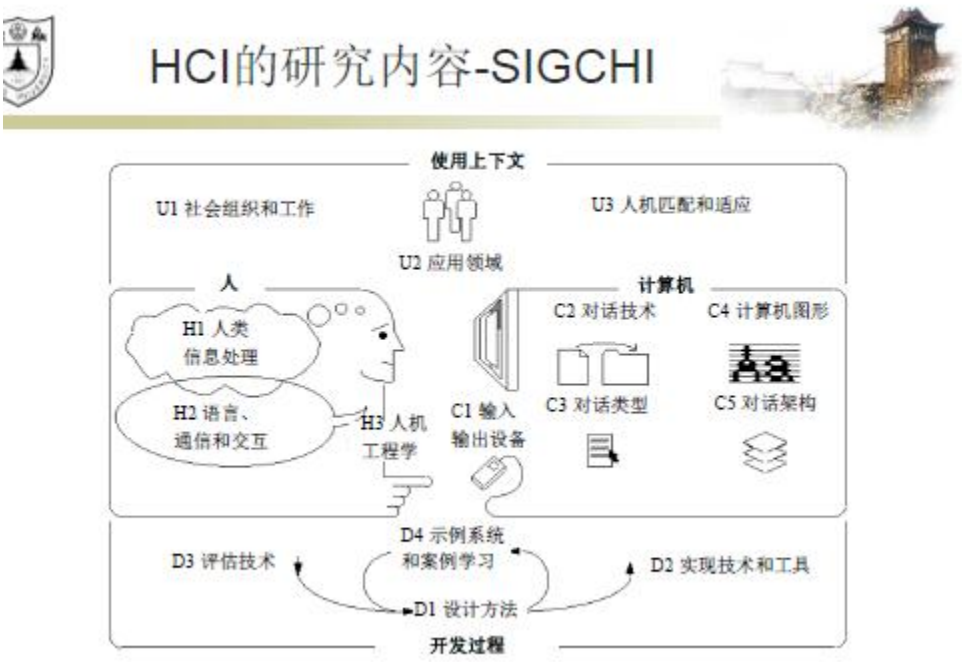
人机交互概述.pdf

人机交互的定义

SIGCHI 定义“HCI is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human”。

老师说要自己理解，得到自己的定义。

人机交互要关注的内容（了解）



人机交互是交叉学科，和软件工程的关系，往往被开发人员忽略

HCI 是个交叉学科

孤立地从一个学科出发不可能设计出有效的交互式系统

Scott Kim 指出

- 学科就像文化，不同学科的人员必须学着尊重对方的语言、习俗和价值观
- 任何能够促进交流的方法都将使设计过程以及最终的产品获益

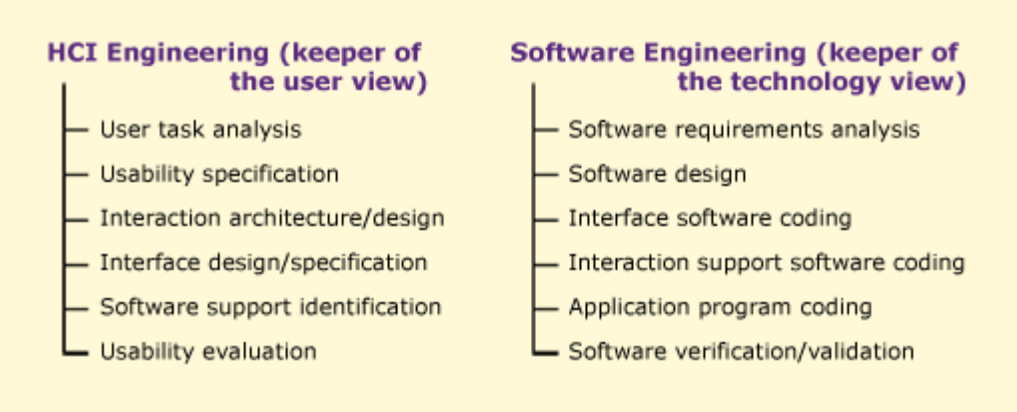
Alan Dix 建议，要特别关注作为核心学科的计算机科学、心理学和认知科学在交互式系统设计方面的应用

交互式系统的分析设计？

人机交互与软件工程

以功能为中心 vs 以用户为中心

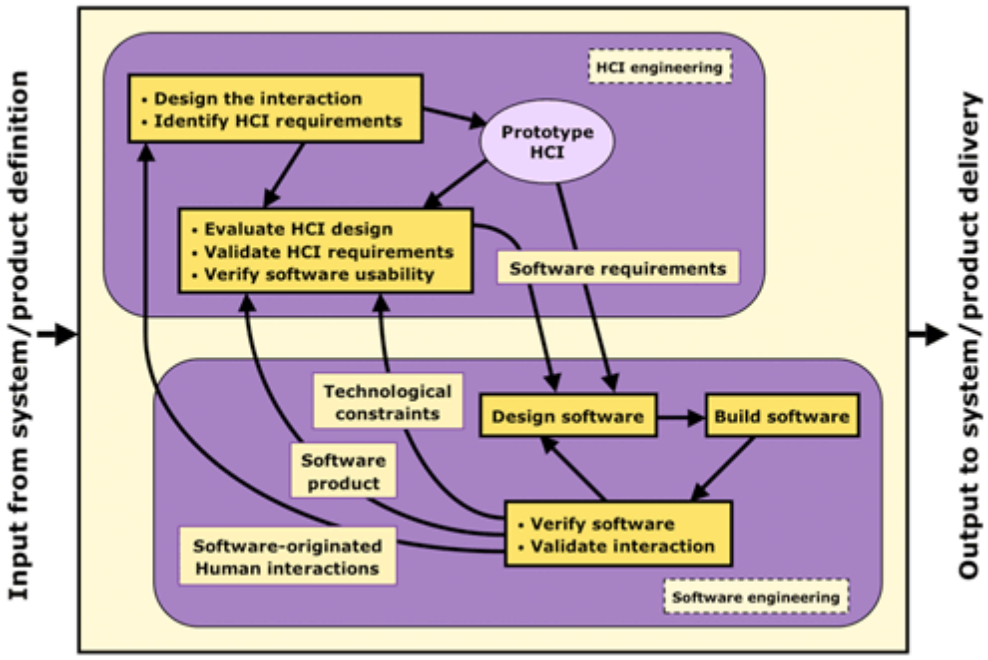
- 二者常常分开讨论



- HCI 促进 SE 因为：

传统 SE 方法在实现交互式系统方面的缺陷

  - 没有提出明确地对用户界面及可用性需求进行描述的方法
  - 不能够在系统开发过程进行中对用户界面进行终端测试
  - 具有完善的系统功能，但是产品的可用性、有效性以及满意度并不高
- 二者在系统工程中关系



- 二者结合的困难

价值观不同

SE: 实施策略和方法选择上常有一定的倾向性

HCI: 包含较多的主观性和灵活性

方法论存在差异

SE: 形式化分析方法

HCI: 非形式化方法

人机交互的基础知识

人机交互的基础知识.pdf

人的特性

人的角度特点：技能

感觉记忆（瞬时记忆）：1 秒；将感知的图片组合成连续的序列，产生动态影像

短时记忆：感觉记忆经编码后形成，保持 30 秒；存储当前正在使用的信息；只能有 7+/-2 个信息单元

7+/-2 理论对交互式设计的影响：菜单 7 个选项；工具栏 7 个图表等；

而事实上：还要依赖人的识别功能、设计要减少对记忆的需求（增加上下文，减少信息单元数目）

长时记忆：短时记忆加工的产物；信息容量几乎无限；要用启发的方式来引导用户完成特定任务，追求创新设计时也要结合优秀交互范型

长时记忆的缺点：遗忘的产生是长时记忆无法提取；容易出错（表面上是用户的错误，实际上是系统设计本身没有很好的引导用户）

视错觉：知觉感受的扭曲，其不可避免；因为对物体的感知和上下文相关

机器的特性

机器：软件硬件设备

文本输入设备

- 键盘：最主要的输入设备；一次响应一个按键；新用户每秒一次，熟练用户每秒 15 次；新手用户可能困哪；小键盘适合移动设备（功能可能受限）
- 和弦键盘：同时响应多个按键，学习时间长
- 投影键盘：内置红色发光器，投影出键盘；红外跟踪手指动作；减小键盘物理空间
- 手写输入：比较自然；慢
- 语音输入：受环境噪音影响；识别效果影响，输入效率低；
- 光学字符识别：让计算机直接读；大批量历史数据的信息化；

定位设备：

- 鼠标：节省空间，可能影响新手用户使用
- 触摸板：反应灵敏，移动速度快；缺点：定位精度低，手指出汗会打滑
- 指點桿：特定是定位精准
- 触摸屏：优点：定位速度快；缺点：精度差，小范围使用困难，成本高
- 尖笔/光笔：较高的定位精度

图形输入设备：

- 扫描仪：平板式（最广泛）；手持式（图像是长条，用于 CAD）；滚筒式（最精密，用于专业印刷排版）
- 数码相机
- 传真机

显示设别：

- CRT 光栅扫描阴极射线管
- 液晶

- 等离子
- 电子墨水
- 发光二极管
- 点字显示器：盲人用

不同的交互范型

- 命令行界面

特点：特定位置键入特定命令方式来执行任务

优点：专家用户可以快速完成；节省资源；动态配置操作选项；键盘操作很精准；支持用户自定义命令

缺点：用户需要有很强记忆能力去掌握命令语言；基于回忆的方式而不是像 GUI 是基于识别的方式，不容易使用；键盘操作出错率搞；要求用户记忆指令的表示方式（违背可用性理论的不让用户了解底层实现）

- 菜单驱动界面

特点：一组层次化菜单来提供可用的功能选项；若干个选项的选择可以改变界面状态

优点：基于识别机制，对记忆的需求较低；自解释性；容易纠错；适合新手用户，快捷键适合专家用户。

缺点：导航方式不灵活；菜单规模大使导航效率降低；占用屏幕空间（下来和弹出式菜单可以节省空间）；对专家用户使用效率不高

- 基于表格的界面

特点：显示给用户是一个表格，里面有需要填写的空格

优点：简化数据输入；只需识别无需学习；适合日常文书处理等需要大量数据键入的工作

缺点：占用大量屏幕空间；导致业务流程较形式

- 直接操纵

直接操纵三个阶段：自由阶段—用户执行操作前的屏幕视图；捕获阶段—用户动作执行过程中屏幕的显示情况；终止阶段—用户动作执行后屏幕的显示情况。

优点：将任务概念可视化，用户辨别方便；容易学习，适合新手用户；基于识别，没有太多记忆要求；支持空间线索，鼓励用户对界面进行探索；可实现对用户操作的快速反馈，具有较高的用户主观满意度。

缺点：实现起来比较困难；对专家用户而言效率不高；不适合小屏幕；对图形显示性能需求高；不具备自解释性，可能产生误导。

- 问答界面

特点： 通过询问用户一系列问题实现人与计算机的交互

Web 问卷是典型的采用问答方式进行组织的应用

应允许用户方便地取消其中一个界面的选项

优点： 对记忆的要求较低；每个界面具有自解释性；将任务流程以简单的线性表示；适合新手用户。

缺点： 要求从用户端获得有效输入； 要求用户熟悉界面控制；纠错过程可能比较乏味。

- 隐喻界面

本质： 在用户已有知识的基础上建立一组新的知识，实现界面视觉提示和系统功能之间的知觉联系，进而帮助用户从新手用户转变为专家用户。

优点： 直观生动； 无需学习

局限性： 不具有可扩展性； 不同用户对同一事物可能产生不同的联想；紧紧地将我们的理念和物理世界束缚在一起； 寻找恰当的隐喻可能存在困难

- 自然语言交互

自然语言的模糊性

受限于理解技术，当前只能够使用受限的语言与计算机进行交流

交互方式，每一种方法的优缺点

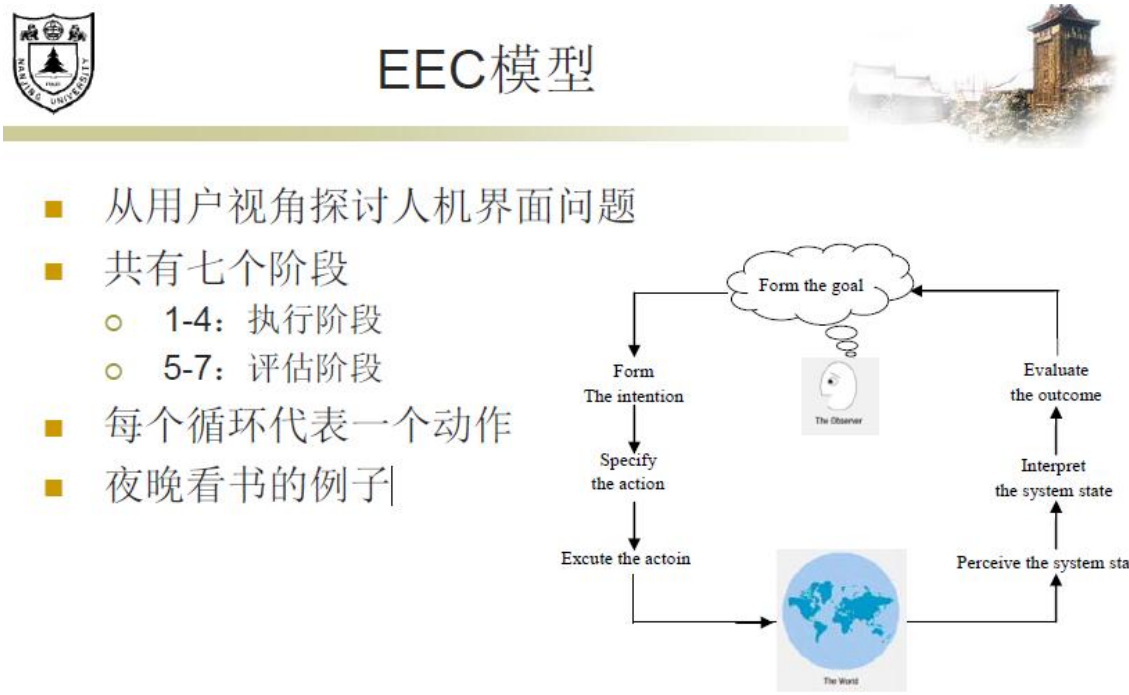
交互框架

执行评估模型（理解），执行隔阂和评估隔阂

框架作用：提供理解或定义某种事物的一种结构；能够帮助人们结构化设计过程；认识设计过程中的主要问题；还有助于定义问题所涉及的领域

EEC 模型：执行/评估活动周期 EEC

定义了活动的四个组成部分
目标(Goal) ≠意图(Intention)：单个目标对应多个意图，每个意图包含一些列活动
执行(Execution)：
客观因素(World)
评估(Evaluation)



执行隔阂

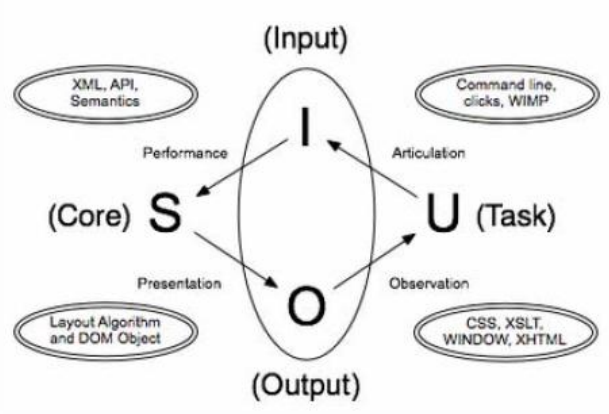
用户为达目标而制定的动作与系统允许的动作之间的差别

评估隔阂

系统状态的实际表现与用户预期之间的差别

扩展EEC模型

- EEC模型不能描述人与系统通过界面进行的通信
- 四个构成部分+四个步骤(翻译过程)
  - 系统：内核语言
  - 用户：任务语言
  - 输入：输入语言
  - 输出：输出语言
- 执行阶段
  - 定义，执行，表现
  - 设计人员应保证从输入到系统的翻译是容易的
- 评估阶段
  - 观察



交互设计原则目标

不仅关心最终产品是否能够满足用户的需要，帮助用户更加高效的完成任务，同时还关注其他诸如用户是否满意以及系统是否容易使用等特性。用"可用性目标"和"用户体验目标"加以区分。

可用性目标

简易可用性工程.pdf

简易可用性工程（5 个可用性指标），记忆力少则易学性损失

可用性目标

- 易学性：指使用系统的难易，即系统应当容易学习，从而用户可以在较短时间内应用系统来完成某些任务；对应学习曲线的开头部分
- 易记性：在使用后应当容易记忆，能快速回想使用方法；影响易记性的因素：位置、组织、惯例、启发；
- 有效率：当学会之后用户应当具有更高的效率，即用户达到学习曲线上平坦阶段时的稳定绩效水平
- 低出错率：保证导致灾难性后果错误的发生率降到最低；错误发生后迅速恢复到正常状态
- 主观满意度：用户对系统的主管喜爱程度

简易可用性工程

特点：以提高产品的可用性为目标的先进的产品开发方法论；借鉴了许多不同领域的方法和技术；强调以人为中心来进行交互式产品的设计研发

可用性度量

常用方法：选择代表目标群体的测试用户去执行一组预定的任务，比较任务的执行情况，针对多维属性

度量一定要针对特定的用户和特定的任务进行

用户对不同任务的可用性结果预期可能不同

因此测试前要明确一组具有代表性的测试任务

易学性度量：找从未使用过系统的用户，统计使用至某种程度的世界

使用效率度量：区分不同的用户群体，统计使用系统的小时数

易记性度量：区分用户；对特定长时间内没有使用系统的用户进行标准用户测试，记录执行特定任务所用的时间；或者对用户进行记忆测试，让其在完成一个特定任务后解释各种命令的作用

错误率度量：通常指不能实现预定目标的操作；统计用户在执行特定任务时统计这种操作的次数

；有可以立即纠正不会对系统带来灾难和不易被用户发现造成危险后果两种

满意度度量：满意度度量都是主观的；询问用户比较合适，将不同人的主观评价综合起来考虑；一般采用问卷评分的方式

举例：图标的可用性度量举例

完整的可用性工程过程

了解用户

竞争性分析

设定可用性目标

用户参与的设计

迭代设计

产品发布后的工作

简化的可用性工程过程

- 用户和任务观察

不要和潜在的用户进行接触；不要满足于间接的接触和道听途说

- 场景（scenario）

使用简便易行的原型工具；通过省略系统的若干部分来减少实现的复杂性；水平原型和垂直原型；

- 简化的边做边说（thinking aloud）

让真实用户在使用的过程中说出内心的所思所想；是最有价值的单个可用性方法；可以了解用户为什么要这么做，并确定其可能对系统产生的误解

- 启发式评估

可以发现很多可用性问题（剩下不能的用边说边做）；避免个人偏见，采用不同的人来进行经验性评估

设计规则

十条启发式规则

系统状态的可见度

系统和现实世界的吻合

用户享有控制权和自主权

一致性和标准化

避免出错

依赖识别而非记忆

使用的灵活性和高效性

审美感和最小化设计

帮助用户识别、诊断和恢复错误

帮助和文档

（曾考给界面，评估设计原则）

8.交互设计过程：生命周期模型，瀑布，螺旋，一个新型模型（强调评估的中心地位，需要用户验证才能进入下一个阶段）

瀑布模型：以文档为中心；不适合交互式软件开发；假设需求是不变的

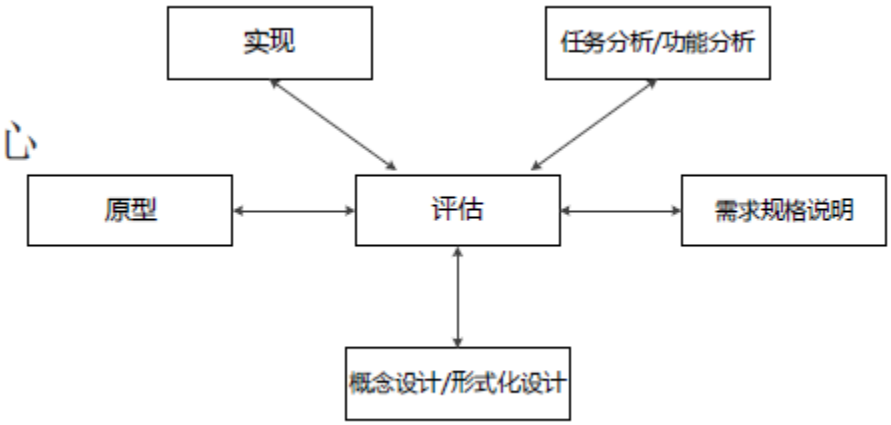
此外传统软件开发模型还有螺旋模型的快速开发模型，这些模型都只适合原则性强的场合，而交互式软件系统的所有需求在开始时是无法确定的

星型模型：

分析模式：自顶向下，组织化，判定和正式化，从系统到用户的方法

合成模式：自底向上，自由思考创造性，这是从用户到系统

特点：没有指定任何活动次序；评估是这个模型的核心；源于开发人员的实际经验



需求

任务角色构建，场景，获取需求（不太容易出），层次化任务分析（HTA，要关注，其文字和图形的格式）

人物角色

与某个系统有关的用户假定的一组公共需要、兴趣、期望、行为模式和责任；这些属性可能是若干用户共有；同一个用户也可以扮演系统的任意个不同角色

获取任务角色

拼凑

采用头脑风暴方法，产生一些零碎概念或模型的片段，先不去考虑他们的细节

组织

将这些片段按照所构造模型的需要进行分组和分类，归并或删除那些冗余重叠的东西

细节

建立和完善相应描述，补充遗漏的数据

求精

对模型进行推敲，以便改进和完善

以上过程循环反复

场景

场景是表示任务和工作构成的“非正式的叙述性描述”

来源：场景说明通常来自专题讨论或者访谈，目的是解释或讨论有关用户目标的一些问题

人物角色+场景剧本→需求

需求定义的 5 个步骤

创建问题和前景综述：简明反应需要改变的情况

头脑风暴：说反话；不受约束

人物角色的期望：要和用户心理模型尽量匹配

构建情境场景剧本：关注人物角色的活动及其心理动机；关注产品如何帮助角色达到目标

确定需要：数据需求；功能需求；其他需求；

迭代的过程

直到需求变得稳定

10.软件设计细节：什么样的软件是好的，品质和特性（了解），如出错处理的对话框，审美，可用性，不同的数量级选择数据库

软件的细节设计

细节设计可用范例

List of Objects：每天遇到，由对象构成的选择表

List of actions：动作为中心，选择项都是动作

List of subject categories（子目录列表）：

List of tools：需要用户对工具有清晰的可以预先知道的认识，通常是线性组织

三种物理结构：多窗口；单一窗口分页；平铺窗口

双面板选择器；

Canvas Plus Palette：画图工具；

单窗口深入：老版 ipod 界面

可选视图：可选视图不光为了装饰，还有结构上的区别；

向导：在界面上一步步引导用户按预定的顺序完成任务；

需要时显示：调色板自定义颜色

有趣的分支：把有趣内容的链接放在一些意想不到的地方并给予好玩的标签来吸引好奇的用户

多级帮助：

设计的折衷

个性化和设置

是否应该让产品具有用户定制功能？

个性化：人们喜欢改变周围事物以适应自己；必须简单易用；在用户确定选择前给他们一个预览的机会；必须容易撤销；

配置：移动、添加或者删除持久对象；富有经验的用户所期望的；包含多种配置形式；

本地化和国际化

国际化：将软件和特定区域、语言脱钩的过程；在移植不同语言和地区时，不对软件内部工程上做修改；只需要做一次

本地化：移植软件时加上特定区域设置有关的信息和翻译文件的过程；为了更适合特定地方的使用而另外增添的特色；针对不同区域做一次；

审美学和使用性

一个漂亮的界面不一定是好界面

审美与实用的冲突：为确保文本可读性，文本背景采用较低对比度，复杂和强烈的对比可能获奖，但不实用

交互设计角度：根据语义和任务因素来进行视觉组织是最重要的；视觉美学的重要性稍低；实现一个良好的基本布局然后再这个基础上改进来实现好的美学效果。

什么样的软件是好的软件？软件设计中的考虑

- 加快系统的响应时间
- 减轻用户记忆负担
- 减少用户等待感（以某种形式的反馈让用户了解操作进行的进度和状态；以渐进方式呈现结果，例如图片显示先全局概括后细节；给用户分配任务；减低用户的期望值）
- 设计好的出错信息：　使用清晰的语言来表达，而不要使用难懂的代码； 使用的语言应当精炼准确，而不是空泛而模糊的； 对用户解决问题提供建设性的帮助； 出错信息应当友好，不要威胁或责备用户

可视化

11.可视化：窗口，菜单（给一个例子，菜单层次不太深）

窗口

多文档界面（photoshop 界面）
优点：　节省系统资源；最小的可视集；协同工作区；多文档同时可视化

缺点：　菜单随活动文档窗口状态变化，导致不一致性；文档窗口必须在主窗口内部

，减弱多文档显示优势； 屏幕显示复杂：子窗口可能在父窗口中被最小化

单文档界面（新版 word 界面）
优点：　从用户角度出发，以文档为中心；界面的视觉复杂性小

缺点：　不能管理分散但相关的文档窗口；相关文档不能从相同类型的

其他文档中分离； 文档打开过多时，任务栏可能被占满

标签文档界面（FireFox 浏览器）

特点：窗口菜单：包含了当前打开窗口的列表
优点：让用户看到哪些窗口是打开的

缺点：不允许用户看到两个及以上的窗口内容

对话框

对话框是典型的辅助性窗口，常用于 1）将某些破坏性的、令人混淆的、不太常用的操作与主要

工作中使用的工具分开考虑，2）通知用户系统的一个错误或潜在的问题，3）没有标题栏图标、状态栏和调整窗口大小的按钮

模态对话框

冻结了它属于的应用，禁止用户做其他操作直到对话中出现的问题被处理；可以切换到其他程序；用户最容易理解，操作非常清晰

应用模态：只停止其所属的应用程序

系统模态：系统中所有程序都停止，多数情况下应用程序不应有系统模态对话框

非模态对话框

打开后无须停止进度，应用程序也不会冻结，其操作范围不确定而难以使用和理解

比如：Word 的查找和替换对话框，画图程序

缺点：缺乏一致的终止命令，如取消、应用、关闭等

按照用途分类

- 属性对话框：呈现所选对象的属性或者设置，并允许用户改变；模态非模态均可；遵循“对象-动词”形式
- 功能对话框：控制单个功能，比如打印，拼写检查等；允许用户开始一个动作，并允许设置动作的细节；配置和实际功能最好隔离开
- 进度对话框：由程序启动而不是用户请求启动；向用户清楚表明：正在运行一个耗时进程，一切正常，剩余时间；并且向用户提供一个取消操作和恢复程序控制的方式
- 公告对话框：GUI 中滥用最多的元素；无需请求由程序启动；有阻塞性公告和临时公告，阻塞性要少用，因为牺牲用户利益而为程序服务；不要用临时对话框作为错误信息框或者确认信息框
- 特殊的对话框（为什么要消除以及如何消除这些特殊的对话框）
    - 错误对话框：用户希望避免错误造成的结果而不是错误消息；错误消息经常存在
    - 消除错误信息：更健壮的软件，为用户提供正面反馈，改进错误消息框（人性化，礼貌，启发性，澄清问题范围，可选择方法，默认程序会做，丢失了哪些信息）
    - 警告对话框：同时用户程序的行为，确认操作赋予用户忽略该行为的权力
    - 消除警告：把提示信息放在主界面中，出现一个注定要关闭的对话框没有必要；软件没有必要阿谀奉承
    - 确认对话框：程序对自己行为的不自信，用对话框来征求许可；把责任推卸给了用户，常规的地方提供确认对话框，可使用户忽略真正的意外；
    - 消除确认对话框：做，不要问；让所有操作可撤销；提供非模板反馈帮助用户避免犯错

管理对话框的内容

取代充斥大量空间的大对话框；拥有更多控件不意味着用户会觉得界面易于使用或功能强大：不同窗格内容必须有放在一起的道理；窗格组织为某个专题上的深度或广度增加

- *标签对话框*
  - 成功的原因是遵循了用户有关“事务存储”的心理模型，即单层分组
  - 比如 PPT 的 option，建议不要堆叠标签（也就是最好是一排标签的形式）
- *扩展对话框*
  - 1990 年左右盛行
  - 新手用户不必面对复杂工具，熟练用户也不必寻找这些工具而烦恼
  - 需要注意设计，最好记住上次调用所处的状态
- *级联对话框*
  - 糟糕的习惯用法，一个对话框的控件在一个层次关系的嵌套中调用另外一个对话框
  - 适合处理深度问题但层次太深导致界面复杂

对话框设计原则

- 把主要的交互操作放在主窗口中
  - 对话框适合主交互流之外的功能
- 视觉上区分模态与非模态对话框
  - 为非模态对话框提供一致的终止命令
- 不要用临时对话框作为错误信息框或确认信息框
  - 保证用户能够阅读
- 不要堆叠标签

常用控件

控件定义： 使用者和数字产品进行交流的屏幕对象；具有可操作性和自包含性

控件的使用必须恰当且合理：大多数布满控件的对话框都不是好的用户界面设计

根据用户目标，控件可分为 4 种基本类型

- 命令控件
  - 接受操作并立即执行
  - 举例：按钮
- 选择控件
  - 允许用户从一组有效选项中选取一个操作数，还能用来设定
  - 举例：
    - 复选框：简单，可见，右牙，确切的文本，占据控件，用户阅读速度慢
    - 列表框：允许用户选择一个，每个文本字符串代表一个命令、对象或属性
    - 下拉列表：列表框的变体

- 显示控件
  - 显示和管理屏幕上信息的视觉显示方式
  - 比如：滚动条（适合用于窗口内容和文本导航器，不适合为无限时间进行导航，因此日历中不适合使用）
- 文本输入控件
  - 让用户输入新的信息而不仅仅是选取；有界输入控件比较好，无界输入会带来灾难

工具栏

微软引入

工具栏是为经常使用的命令设置的，对新手用户帮助不大；工具栏是单行(或单列)排列且始终可见的图形化立即菜单项

工具栏 vs 菜单

工具栏主要为常用功能提供快速访问

图形的表意特征较文本更适合担当这种角色

Q:为什么不能同时使用文本和图像？

屏幕空间非常宝贵

解决：工具提示

解释工具栏控件

Apple 公司的 System 7 操作系统最早尝试解决该问题：气球帮助

工具提示(ToolTips)

没有理由将工具栏控件限制为图标按钮

组合框：Word 样式、字体和字号控制；状态指示；工具栏上的菜单； 可移动工具栏

可定制工具栏； 带条

菜单

菜单是访问系统功能的工具，已经成为窗口环境的标准特征

- 某些功能可以立即执行，或由选择的菜单命令激活一个包含相关功能的对话框
- 菜单适合初学者，包含完整工具集合
- 不仅仅适用于初学者

菜单必不可少的组成部分

- 菜单标题
- 菜单选项
- 最重要的特性：描述性、一致性

菜单栏

- 所有窗口必备的基本组件
- 菜单栏是代表下拉式菜单的菜单
- 菜单选项的标签、位置、归类等均已标准化：从左向右：文件、编辑、视图和窗口，帮助位于窗口的最右边；可选菜单常出现在编辑和窗口菜单之间

菜单注意事项

- 菜单应该按语义及任务结构来组织：糟糕的例子：File 菜单
- 合理组织菜单接口的结构与层次：菜单太多或太少都表明菜单结构有问题
- 菜单及菜单项的名字应符合日常命名习惯
- 菜单选项列表即可以是有序的也可以是无序的: 频繁使用的菜单项应当置于顶部
- 为菜单项提供多种的选择途径:尽可能使用工业标准
- 对菜单选择和点取设定反馈标记：如灰色屏蔽，为选中的菜单项加边框在菜单项前面加√符号等

用户界面设计原理

结构原理；简单性原理；可见性原理；反馈原理；重用原理；宽容原理

模型和理论

预测模型是重点，GOMS，KLM，Fitts

预测模型

能够预测用户的执行情况，但不需要对用户做实际测试

特别适合无法进行用户测试的情况

不同模型关注用户执行的不同方面

GOMS 模型

是关于人类如何执行认知—动作型任务以及如何与系统交互的理论模型

采用“分而治之”的思想，将一个任务进行多层次的细化

把每个操作的时间相加就可以得到一项任务的时间

操作指用户的目光从屏幕的一处移到另一处、识别出某个图标、手移到鼠标上

为什么叫 GOMS？

- Goal-目标： 用户要达到什么目的
  - 通常是层次化的，通常表示为动作-对象序列（比如复制-文件）
- Operator-操作：用户为了完成任务必须执行的基本动作
  - 有外部操作和心理操作两种
  - 操作时间是上下文无关的，花费的时间与用户正在完成什么样的任务或当前的操作环境没有关系
- Method-方法：为实现目标需要的操作序列
  - 外部+心理

```
GOAL: CLOSE-WINDOW
.   [select GOAL: USE-MENU-METHOD
            .   MOVE-MOUSE-TO-FILE-MENU
            .   PULL-DOWN-FILE-MENU
            .   CLICK-OVER-CLOSE-OPTION
            GOAL: USE-ALT-F4-METHOD
            .   PRESS-ALT-F4-KEYS]
```

- Selection-选择规则：选择规则是用户要遵守的判定规则
  - 确定在特定环境下所使用的方法
  - GOMS 中并不认为这是一个随机的选择

GOMS 实例

任务: 编辑文档

```
GOAL:EDIT-MANUSCRIPT
   GOAL:EDIT-UNIT-Task … repeat until no more unit tasks
    GOAL:ACQUIRE-UNIT-TASK
       GET-NEXT-PAGE … if at end of manuscript
       GET-NEXT-TASK
    GOAL:EXECUTE-UNIT-TASK
       GOAL:MODIFY-TEXT
          [select: GOAL:MOVE-TEXT… if text is to be moved
                   GOAL:DELETE-PHRASE… if phrase is to be deleted
                   GOAL:INSERT-WORD] … if a word is to be inserted
             VERIFY-EDIT
```

GOMS 方法步骤

- 选出最高层的用户目标
- 写出具体的完成目标的方法：即激活子目标
- 写出子目标的方法：递归过程，一直分解到最底层操作时停止
- 子目标的关系

GOMS 模型分析

优点：能够容易地对不同界面或者系统比较分析；

局限：假设用户完全按照一种正确的方式进行人机交互，没有清楚地描述错误处理的过程；只针对那些不犯任何错误的专家用户；任务之间的关系描述过于简单；忽略了用户间的个体差异

四种 GOMS 模型

击键层次模型：简化的设计

对用户执行情况进行量化预测

用途：预测无错误情况下专家用户在下列输入前提下完成任务的时间；便于比较不同系统；确定何种方案能最有效地支持特定任务；

基本操作符：K 按键；P 定位；H 复位；D 绘图；M 思维；R 系统反应时间

执行时间预测方式：列出操作次序，累加每一项操作的预计时间

CMN COMS：伪代码描述，结构严格

NGOMSL：程序行使，结构泛化，仅能够预测性能和学习次数

CPM GOMS:基于 Model-Human Processer；允许操作符的并行操作

KLM 分析

GOMS 只考虑了时间，但是没有考虑错误啊，学习性啊等等问题

KLM 是为了在交互设计早期阶段为用户性能提供有效、准确的模型

Fitts 定律

用户访问屏幕组件的时间对于系统的使用效率是非常重要

Fitts 定律描述了人类运动系统的信息量：

$$C = B \log_2(S/N+1)$$

C 是有效信息量(比特)，B 是通道带宽　S 映射为运动距离或振幅(A)，N 映射为目标的宽度(W)

三个部分：

困难指数 ID：$\log_2(A/W+1)$ (bits)

运动时间 MT：$a + b*ID$　Mac 改写 a+b log2(A/W+1)

性能指数 IP：ID/MT

a, b 的确定：

设计一些列任务；对每一种条件下的任务：尝试多次、记录每次执行时间、进行统计分析；记录准确性：记录选择的 x.y 坐标或者鼠标落在目标之外的百分比

Fitts 定律建议

- 大目标、小距离具有优势
  - 对选择任务而言，其移动时间随到目标距离的增加而增加，随目标的大小减小而增加
- 屏幕元素应该尽可能多的占据屏幕空间
- 最好的像素是光标所处的像素
- 屏幕元素应尽可能利用屏幕边缘的优势
- 大菜单，如饼型菜单，比其他类型的菜单使用简单

Fitts 应用

1. 缩短当前位置到目标区域的距离：比如右键菜单技术
2. 增大目标大小以缩短定位时间
3. Mac OS dock 设计

交互设计的评估框架

12.交互评估框架（深入理解）:实验设计，目标（可用性标准），不要大而全，考察核心；任务的制定，是可执行的任务，例如：要明确添加的歌曲名称

评估的目标

评估系统功能的范围和可达性；评估交互中用户的体验；确定系统的某些问题；

评估原则

评估应该依赖于产品的用户；评估与设计应该结合进行；评估应在用户的实际工作任务和操作环境下进行；要选择有广泛代表性的用户

不同的评估泛型对应着不同的技术

评估泛型：快速评估；可用性测试；实地研究；预测性评估

评估技术：观察用户；查询用户意见；查询专家意见

Decide 评估框架

六个步骤

- 决定评估需要完成的总体目标

    - 为什么要评估？
        - 产品设计是否了解用户需要？界面是否满足一致性要求？etc
    - 举例
        - 设计界面是需要量化评价界面质量
        - 为儿童设计新产品要使产品吸引人
- 发掘需要回答的具体问题
    - 问题可以逐层分解
- 选择用于回答具体问题的评估范型和技术
    - 泛型决定了技术的类型
    - 必须权衡实际问题和道德问题
    - 可以结合使用多种技术
- 标识必须解决的实际问题，如测试用户的选择
    - 选择恰当的用户参与评估
    - 设备和设施的安排
    - 期限和预算
    - 是否需要专门技能
- 决定如何处理有关道德的问题
    - 个人隐私
    - 指导原则：说明研究目的以及参与者工作；说明保密事项，对用户和项目；测试对象是软件而非个人
- 评估解释并表示数据
    - 搜索什么类型的数据，如何分析，如何表示
    - 可靠性
    - 有效性
    - 偏见
    - 范围
    - 环境影响

可用性问题

评估结果总是可用性问题清单以及改进建议

方法：

- 基于量化数据的分级
- 问题严重性的主观打分，取平均值
- 可用性分级的两个因素
- 该问题只在第一次使用时出现，还是会永远出现

评估用户测试

用户测试：在受控环境中（类似于实验室环境）测量典型用户执行典型任务的情况；目的是获得客观的性能数据，从而评价产品或系统的可用性，如易用性、易学性等； 最适合对原型和能够运行的系统进行测试；可对设计提供重要的反馈；在可用性研究中，往往把用户测试和其他技术相结合

1. 定于目标和问题
2. 选择参与者：相同参与者；不同参与者；参与者配对
3. 设计测试任务：任务与目标相关
4. 明确测试步骤：测试前准备好进度表和说明，设置好各种设备；正式测试前进行小规模测试
5. 数据搜集：常用度量有完成任务时间，停止使用一段时间后测试的完成时间，单位时间内出错数
6. 数据分析：自变量和因变量

评估之观察

Basic

观察用户怎样工作（因为用户不能客观完整地描述产品使用情况；观察法是可用性方法中最简单啊的；适合产品开发的任何阶段）

两种观察方式：真实环境的观察；受控环境的观察

实验室观察

观察框架

观察过程发生的事件都非常复杂且变化迅速

观察框架用于组织观察活动和明确观察重点

Goetz and Lecomfte 框架

关注事件的上下文、涉及的人员和技术

人员：有哪些人员在场？他们有何特征？承担什么角色？

行为：人们说了什么？做了什么？举止如何？是否存在规律性的行为？语调和肢体语言如何？

时间：行为何时发生？是否与其他行为相关联？

地点：行为发生于何处？是否受物理条件的影响？

原因：行为为何发生？事件或交互的促成因素是什么？不同的人是否有不同的看法？

方式：行为是如何组织的？受哪些规则或标准的影响？

Robson 框架

有助于组织观察和数据搜集活动

空间：物理空间及其布局如何？

行为者：涉及哪些人员？人员详情？

活动：行为者的活动及其原因？

物体：存在哪些实际物体（如家具）？

举止：具体成员的举止如何？

事件：所观察的是不是特定事件的一部分？

目标：行为者希望达到什么目标？

感觉：用户组及个别成员的情绪如何？

Tips：观察中要让用户边做边说；合作评估：让两位用户合作，相互讨论（这样提供更多信息）；

现场观察

- 明确初步的研究目标和问题
- 选择一个框架指导观察
- 决定数据记录方式
  - 笔记、录音、摄像，还是三者结合
  - 确保设备到位并能正常工作
- 评估后，尽快与观察者或被观察者共同检查所记录的笔记和其他数据
  - 研究细节，找出含糊之处
  - 人的记忆能力是有限的，最好 24 小时内回顾数据

定性分析：常用方法：找出关键事件；内容分析；会话分析；话语分析

定量分析：统计分析；

观察与访谈相结合

观察方法只能展示用户做了什么，而无法知道用户为什么这样做

访谈可以请用户详细讲述记录里面任何可能引发可用性问题的地方

如对一个没用过系统某个功能的用户，询问为什么没有使用某项功能

让用户面对记录数据时应非常小心

避免让用户产生被监视的想法

评估之询问

13 专家的评估技术：评估技术的组合，常见组合方式

了解用户的需要和对产品的建议

观察用户，询问用户，请专家帮忙

访谈

有目的的对话过程

类型：开放式；结构化访谈；半结构化访谈；集体访谈

指导原则

- 避免过长的问题；
- 避免使用复合句；
- 闭眼使用可能让用户感觉尴尬的术语或他们无法理解的语言
- 避免使用有诱导性的问题
- 尽可能保证问题是中性的

开始阶段（自我介绍，访谈原因）→热身阶段（简单问题）→主要访谈（逻辑上从易到难）→冷却阶段（若干容易问题）→结束访谈（感谢受访者）

焦点小组的方式访谈，6-9 人小组，在使用界面一段时间后的用户需要和感受

问卷调查

问卷设计原则

- 应确保问题明确，具体
- 在可能时，采用封闭式问题并提供充分的答案选项
- 对于征求用户意见的问题，应提供一个“无看法”的答案选项
- 注意提问次序，先提出一般化问题，再提出具体问题
- 避免使用复杂的多重问题
- 在使用等级标度时，应设定适当的等级范围，并确保它们不重叠
- 避免使用术语
- 明确说明如何完成问卷
- 在设计问卷时，既要做到紧凑，也应适当留空

和访谈的区别

问卷调查或访谈都属于间接方法，因为两者都不对用户界面本身进行研究，而只是研究用户对界面的看法。 都不能完全听信和采纳用户的说法

访谈形式更自由；难以获得确切数据； 需要花费更多时间；可在访谈后立即得到结果； 可能回避某些“敏感问题”的真实想法

认知走查

优点：

不需要用户参与

不需要可运行的原型

能找出非常具体的用户问题

专注于用户任务；能够产生定量数据

符合参与式设计原则

缺点

工作量大，非常费时

关注面有限： 只适合于评估一个产品的易学习性；不太容易发现使用效率方面的可用性问题

需要各方面的专家，速度慢

由于时间限制，通常只能评估有限的场景

启发式评估

灵活运用

简介

一种灵活而又相当廉价的评估方式

介绍阶段：告诉专家们需要做什么； 若评估可运行的产品，则评估人员需要了解具体的用户任务以便明确研究重点

评估阶段：专家采取“角色扮演”的方法，模拟典型用户使用产品的情形，从中找出潜在的问题 ；记录潜在可用性问题和严重程度

总结阶段

七项启发式原则

内部一致性：用户能明确所有措辞的含义

对话的简单性： 对话不应包含无关、不必要或极少用到的信息

快捷链接： 适合无经验和有经验用户使用

尽可能减轻用户的记忆负担：不应要求用户记忆先前的对话信息

预防错误：能够预防错误的发生

提示信息： 应能掌握系统的运行状态

内部控制：允许用户立即退出错误选择

复习：Nielsen 的十条启发式规则

系统状态的可视性

系统应与真实世界相符合

用户的控制权及自主权

一致性和标准化

帮助用户识别、诊断和修复错误

预防错误

依赖识别而非记忆

使用的灵活性及有效性

最小化设计

帮助及文档

16.以用户为中心的思想的反思（如何解决问题），以活动为中心的思想（寻找匹配的平衡点）

17.考试：简答题：5-6 道

执行隔阂评估隔阂是什么，有什么意义，

不会考什么人机交互

小问答：具体界面，给出交互设计原则（1-2 局话答道点上）

给出应用背景（2-3 道）

大题（1 题）：设计一个菜单，给出评估方法，设计实验