# 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

问答题：2-3

具体界面，给出交互设计原则（1-2句话答道点上）

给一个问卷，原型找问题

给出应用背景

大题（1题）

设计一个菜单，给出评估方法，设计实验。

# Chapter1 概述

## HCI研究内容——CHUD

## HCI与哪些学科有关联：

* **心理学与认知科学；**人体工程学；社会学；**计算机科学和工程学**；商务知识；图形设计；科技写作；产品设计，工业设计，电影产业……

## HCI vs.SE：

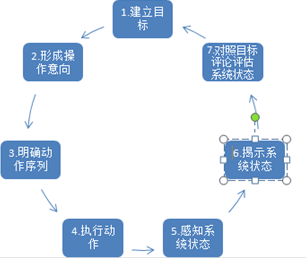
* 关注点和评估方式不同
  + 以功能为中心 vs. 以用户为中心
* 传统SE方法在实现交互式系统方面的缺陷
  + 没有提出明确地对用户界面及可用性需求进行描述的方法
  + 不能够在系统开发过程进行中对用户界面进行终端测试
  + 具有完善的系统功能的产品，可能可用性、有效性以及满意度并不高
* 价值观不同——方法论存在差异

# Chapter2基础

## 执行/评估活动周期EEC

活动定义为四个基本组成部分：

* 目标（Goal），想做什么，这是最重要的；
* 执行（Execution），要实现目标必须进行的操作；
* 客观因素（World），执行活动时必须考虑的客观条件；
* 评估（Evaluation），用于衡量活动执行的结果和目标之间的差距。



七个阶段可归纳为两个步骤：执行（1-4）和评估（5-7）

## 隔阂

* **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。

## 命令行交互

（用户通过在屏幕某个位置上键入特定命令的方式来执行任务）

* 优点
  + - * 专家用户能够快速完成任务；
  + 较GUI节约系统资源；
  + 可动态配置可操作选项；
  + 键盘操作较鼠标操作更加精确；
  + 支持用户自定义命令
    - 缺点
  + 基于回忆的方式——记忆能力高要求
  + 键盘操作，出错频率较高；

## 菜单驱动界面

（以一组层次化菜单的方式提供用户可用的功能选项，一个或多个选项的选择可以改变界面的状态）

* + - 优点
  + 基于识别机制，对记忆的需求较低；
  + 具有自解释性；
  + 容易纠错；
  + 适合新手用户。若提供了较好的快捷键功能，则对于专家用户同样适用。
    - 缺点
  + 导航方式不够灵活；
  + 当菜单规模较大时，导航效率不高；
  + 占用屏幕空间，不适合小型显示设备。
  + 对专家用户而言使用效率不高

## 基于表格的界面

（显示给用户的是一个表格，里面有一些需要用户填写的空格）

* + - 优点
  + 简化数据输入；
  + 只需识别无需学习；
  + 特别适合于键入大量数据的工作
    - 缺点
  + 占用大量屏幕空间；
  + 导致业务流程较形式

## 直接操纵——所见即所得

（该系统展现了真实世界的一种扩展，对象和操作一直可见，迅速且伴有直观的显示结果的增量操作，增量操作可以方便地逆转）

* 自由阶段——指用户执行操作前的屏幕视图；
* 捕获阶段——在用户动作（点击、点击拖拽等）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，用户可以非常方便地辨别他们；
  + 容易学习，适合新手用户；
  + 基于识别，对记忆的要求不高，可减少错误发生；
  + 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；
  + 不适合小屏幕显示设备；
  + 对图形显示性能的需求较高；
  + 不具备自解释性，可能误导用户

## 问答界面——向导

(通过询问用户一系列问题实现人与计算机的交互)

* + - 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（Metaphor）界面

（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——siri

* + - 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

# Chapter3 简易可用性工程

HCI的目标

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。

## 可用性目标：

1. 易学性（learnability）：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，是最基本的可用性属性。
2. 有效率（efficiency）：当用户学会使用产品之后，用户应该具有更高效率

效率指熟练用户到达学习曲线上平坦阶段时的稳定绩效水平

1. 易记性（memorability）：用户在学会使用软件后应当容易记忆；学会某个系统后，应能够迅速回想起它的使用方法

易记性的影响因素：

* 位置：将特定对象放在固定位置
* 分组：对事物按照逻辑进行恰当的分组
* 惯例：尽可能使用通用的对象或符号
* 冗余：使用多个感知通道对信息进行编码

1. 低出错率（errors）：人是会犯错误的，有些错误会被用户发现并纠正，有些错误会带来灾难性后果。

* 保证导致灾难性后果错误的发生频率降到最低
* 保证错误发生后迅速恢复到正常状态

1. 主观满意度（satisfaction）：

* 用户对系统的主观喜爱程度
* 某些情况下，系统的娱乐价值比完成任务的速度更为重要，如家用计算、游戏等非工作环境的系统

## 用户体验目标：

什么是用户体验：

* 用户在与系统交互时的感觉——投其所好
* 较可用性目标主观——可用性可能对用户体验带来阻碍

## 可用性度量：

常用方法：

* 选择测试用户
* 使用系统执行一组预定的任务
* 比较执行情况
* 针对多维属性
* 取每个可用性属性的平均值
* 查看整体分布情况
* 度量一定要针对特定的用户和特定的任务进行

不同用户对不同任务（规模，目的…）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用户和测试任务

### 易学性度量：

* 可用性属性中最容易度量的属性
* 找一些从未使用过系统的用户
* 能够代表系统的目标用户
* 区分计算机新手用户和熟练用户
* 统计他们**学习**使用系统直至达到某种熟练程度的**时间**
* 特定熟练程度
* 用户能够完成某个特定的任务
* 或用户能够在特定的时间内完成一组特定任务

### 高效性度量：

当发现用户的绩效水平在一段时间内不再提高时，就认为已经达到了该用户的稳定绩效水平

首先定义熟练水平；然后找到满足此水平的用户；让其完成特定任务，记录时间，以表示熟练使用系统的效率！

### 易记性度量：

* 用户分类
* 新手用户，熟练用户，非频繁使用用户
* 对非频繁使用用户进行测试最能体现系统的易记性
* 度量方法
* 对在特定长时间内没有使用系统的用户进行标准用户测试
* 记录下这些用户执行特定任务所用的时间
* 对用户进行记忆测试
* 如在用户完成一个应用系统的特定任务后，让用户解释各种命令的作用

### 错误率度量：

* 错误——通常指不能实现预定目标的操作
* 度量
* 在用户执行特定任务时通过统计这种操作的次数
* 可以在度量其他可用性属性的同时来度量
* 错误分类
* 错误发生后能够被用户立刻纠正，不会对系统带来灾难性的影响
* 不易于被用户发现，从而可能造成最终结果存在问题
* 只在首次出现，还是一直出现。

### 满意度度量：

* 满意度度量评价都是主观的
* 以询问用户的方式进行度量更合适
* 为减少单个用户评价的主观性，把多个结果综合起来取其平均值
* 度量通常在用户测试完成后进行
* 要求用户通过简单的调查问卷对系统打分
* 可以1-5或1-7的Likert度量尺度或语义差异尺度作为打分标准
* 一定要在用户使用系统执行真实的任务之后再询问他们的看法

## 可用性工程的四种主要技术：

1. 用户和任务观察

* 了解产品的目标用户是可用性工程的第一个步骤
* 要直接与潜在用户进行接触——不要满足于间接的接触和道听途说

1. 场景

* 简便易行的原型工具
* 水平原型：减少功能的深度并获得界面的表层
* 垂直原型：减少功能的数量而对所选功能进行完整实现

1. 边做边说法

* 让真实用户在使用系统执行一组特定任务的时候，讲出他们的所思所想
* 实验人员需要不断地提示用户，或请他们事先观摩

1. 启发式评估

* 研究表明，能够发现许多可用性问题
* 为避免个人的偏见，应当让多个不同的人来进行经验性评估

## 交互设计规则

* 可学习性——新用户能用它开始有效的交互并能获得最大的性能
* 灵活性——用户和系统能以多种方式交换信息
* 健壮性——在决定成就和目标评估方面对用户提供的支持程度

### Shneiderman的八条“黄金规则”

1. 尽可能保证一致

 2. 符合普遍可用性

 3. 提供信息丰富的反馈

 4. 设计说明对话框以生成结束信息

 5. 预防并处理错误

 6. 让操作容易撤销

 7. 支持内部控制点

 8. 减轻短时记忆负担

### Nielsen的十项启发式规则：（重要）

**交互**

* 系统状态的可见度——系统要经常给用户反馈
* 使用的灵活性和高效性——系统允许自定义和配置

**静态设计**

* 系统和现实世界的吻合——合情合理。
* 一致性和标准化——尊重惯例
* 审美感和最小化设计——简洁清晰专注
* 依赖识别而非记忆——自解释，清晰可见

**错误处理**

* 避免出错
* 用户享有控制权和自主权——可以摆脱错误环境
* 帮助用户识别、诊断和恢复错误
* 帮助和文档

# Chapter4 设计过程

## 交互设计过程的基本过程：

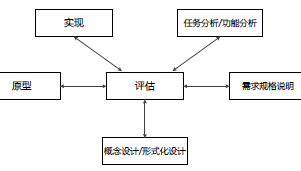
* 标识用户需要并建立需求
* 谁是目标用户
* 交互式产品应提供哪些支持
* 开发满足需求的候选设计方案
* 设计的核心活动
* 概念设计和物理设计
* 构建设计的交互式范本
* 评价设计的最佳方法就是让用户与产品交互
* 不一定是可运行的软件版本
* 评估设计
* 评估它的可用性和可接受性
* 制定各种评估标准

## 交互设计过程的问题（了解）：

* 如何选取用户
* 用户与涉众
* 如何明确需求
* 需求、问题和目标
* 候选方案
* 局限性：人员，已有产品。。。。
* 最终方案的决策
* 以用户为中心。

## 设计过程的生命周期：

### 星型生命周期模型



* 分析模式
* 自顶向下、组织化、判定和正式化，它是从系统到用户的方法
* 合成模式
* 自底向上、自由思考、创造性，这是由用户至系统的方法

## 合理的以用户为中心的交互设计，LUCID

预想——发展清晰、共享的产品场景，使概念草案具体化

发现——研究用户以决定高端的用户需求、术语和智力模型

设计基础——发展概念设计，创造关键屏幕画面原型

设计细节——将高端设计加以充实，形成完备的详细说明书

构建——通过回顾和后期改变管理方式来支持生产过程

发布——通过有力的推广来支持用户向新产品的过渡；进入最后的可用性测试

# Chapter5 需求

## 人物角色的构造

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

是基于观察到的那些真实人的行为和动机，并且在整个设计过程中代表真实的人

是在人种学调查收集到的实际用户的行为数据的基础上形成的综合原型

基于如下问题：

* 谁将使用系统？
* 这些用户属于哪些类型的人群？
* 是什么因素决定他们将怎样使用系统？
* 他们与软件的关系有什么特征？
* 他们通常需要软件提供什么支持？
* 他们对软件会有怎样的行为？他们对软件的行为有什么期望？

过程：

 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

场景

是表示任务和工作构成的“非正式的叙述性描述”。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（具体例子见PPT chapter5）**

**** 创建问题和前景综述

 头脑风暴

 人物角色的期望

 构建情境场景剧本

 确定需要

## 任务分析：

层次化任务分析（HTA）是应用最广的任务分析技术：把任务分解为若干子任务，再把子任务进一步分解为更细致的子任务。之后，把他们组织成一个“执行次序”，说明在实际情形下如何执行各项任务

任务分析是一个迭代过程——终止点：

* 任务包含了复杂机械响应的地方，如鼠标移动，此时分解没有价值
* 涉及内部决断的地方，若决断和查找文档等外部动作相关，则分解；若为纯粹认知性，则终止

# Chapter6 HCI细节

## 设计框架：

六个步骤：

* 定义外形因素和输入方法
* 定义功能和数据元素
* 决定功能组合层次
* 勾画大致的设计框架
* 构建关键情景剧本路径
* 通过验证性的场景剧本来检查设计

## 细节设计可用模式

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

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（可结合可用性目标）

* 加快系统的响应时间——良好的利用空余时间进行预测计算。
* 减轻用户的记忆负担：回忆用户上次的行为预测用户可能的操作
* 减少用户的等待感——进度可观察；分阶段显示
* 设计良好的出错信息

# Chapter7 可视化设计

界面的主要组成部分：WIMP（窗口，图标，菜单，指点设备）

## 窗口（Window）：

状态：

* 最大化
* 最小化
* 还原
* 平铺(Tile)窗口：允许拖放操作
* 重叠(Overlapping)窗口：有效利用屏幕空间
* 层叠(Cascade )窗口：可视化组织各窗口

窗口界面类型：

* 多文档界面（Photoshop）
* 单文档界面（word）
* 标签文档界面（浏览器）

## 菜单（Menu）：

* 访问系统功能的工具，已经成为窗口环境的标准特征
* 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
* 设计菜单时应遵循的原则：
* 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回馈标记
* 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

## 对话框：

* 典型的辅助性窗口
* 常用于将某些破坏性的、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
* 通知用户系统的一个错误或潜在的问题
* 没有标题栏图标、状态栏和调整窗口大小的按钮
* 分类：模态对话框vs. 非模态对话框
* 模态对话框：
* 冻结了它属于的应用，禁止用户做其他操作，直到处理了对话框中出现的问题
* 可以切换到其他程序进行操作
* 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”
* 使系统中的所有程序都停止
* 大多数情况下，应用程序不应该有系统模态对话框
* 非模态对话框：
  + - 打开后无须停止进度，应用程序也不会冻结
    - 由于其操作范围不确定而难以使用和理解
    - 举例
* Word的查找和替换对话框
* 画图程序
* 可以在主窗口和非模态对话框之间拖动对象
* 存在的问题：缺乏一致的终止命令，如取消、应用、关闭等
* 对话框的不同目的
* 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
* 控制选择遵循“对象-动词”形式
* 选择对象，通过属性对话框为所选对象选择新的设置
* 功能对话框：
* 控制单个功能，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，如打印多少页、多少份、哪一台打印机
* 注意：
* 一个功能可以配置，不意味着用户每次调用都想配置
* 配置和实际功能最好隔离开
* 进度对话框
* 由程序启动，而不是根据用户请求启动
* 向用户清楚地表明正在运行一个耗时的进程
* 向用户清楚地表明一切正常
* 向用户清楚地表明进程还需多长时间
* 向用户提供一种取消操作和恢复程序控制的方式
* 进度表应表明相对整个过程所耗费时间的进度，而不是相对整个过程规模的进度
* 公告对话框
* GUI中滥用最多的元素
* 无须请求，由程序直接启动
* 阻塞型公告VS. 临时公告
* 错误、警告和确认消息都是阻塞型公告对话框

为程序服务，牺牲用户利益，应尽量避免

* 绝不要用临时对话框作为错误信息框或确认信息框！
* 管理对话框内容：
* 标签对话框（如选项对话框）
* 取代充斥大量控件的大对话框
* 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
* 成功的原因是其遵循了用户有关“事物存储”的心理模型，即单层分组
* 不同控件组成多个平行窗格，只有一个层次深度
* 扩展对话框（如查找与替换对话框）
* 新手用户不必面对复杂的工具，熟练用户也不必为寻找这些工具烦恼
* 注意：
* 必须小心设计
* 否则对初学者傲慢无礼，专家使用也有麻烦
* 最好记住上次被调用时所处的使用状态
* 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一个对话框
* 分析
* 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

## 控件

* 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（何时用何种控件）
* 命令控件
* 选择控件
* 显示控件
* 文本输入控件
* 命令控件：
* 接收操作并立即执行
* 举例：按钮
* 单击释放后立即执行
* “可按压特性”，点击时视觉发生改变
* 不改变会使用户不安，“到底被按下了吗？”
* 图标按钮（butcons）：工具栏中的按钮
* 矩形->方形，有文字->有图形，有突起->无突起
* 选择控件
* 允许用户从一组有效的选项中选择一个操作数，还能用来设定操作
* 也可用来定义形容词或副词
* 部分情况下，选择控件也可以触发操作
* 如使用下拉列表调整字体大小等，较为自然
* 网页中下拉列表触发操作有可能把用户搞糊涂
* 常见选择控件
* 复选框



简单、可见、优雅

确切的文本使复选框清楚明确

但用户不得不放慢阅读速度

占据了数量可观的屏幕空间

方形复选框已经成为一个重要的标准

变体：单选按钮

* 列表框



* 允许用户从有限的文本字符串中选择，每个文本字符串代表一个命令、对象或属性
* 又称“列表框”或“列表视图”
* 变体
* 下拉列表、列表中的图标、预览视图
* 标准的选择模态是相互排斥的
* 选择一个事物时，先前选定的事物就会取消
* 下拉列表
* 显示控件
* 显示和管理屏幕上信息的视觉显示方式
* 典型例子：滚动条，屏幕分割线
* 输入控件



* 让用户在程序中输入新的信息，而不仅仅是从列表中选择
* 无界输入可能导致十分严重的问题
* 有界输入控件
* 用于在任何需要有数值界限的地方
  + 杜绝“实际上不能却说可能”的粗鲁方式
* 举例：滑动条、微调控制项、标尺等

## 工具栏

* 工具栏VS. 菜单
* 都提供对程序功能的访问
* 菜单提供完整的工具集，主要用于教学
* 工具栏是为经常使用的命令设置的，对新手用户帮助不大
  + 工具提示可以在一定程度上缓解这个问题
* “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”
  + - 工具栏上的图标与文本
* 文本标签精确，但阅读与识别速度较慢
* 工具栏和菜单的用途不同
  + 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
* 图标图像
* 找代表事物的图像比寻找代表动作或关系的图像容易得多，如垃圾桶、打印机等

## 用户界面设计原理

**（例子映射规则）**

1. 结构原理（根据清楚而一致的模型，以一种有意义和有用的方式对用户接口进行组织，把相关的东西放在一起，把不相关的东西分开放，区别不同的东西，使类似的东西看起来相似。）
   * + 好的用户界面设计对用户界面的组织方式应符合所支持工作的结构，符合用户对工作的认识
     + 过滥地运用隐喻会让用户界面难以理解
2. 简单性原理（使简单、常用的功能简便易行，用用户自己的语言进行简明易懂的交流，对冗长的操作过程提供与其语义相关的快捷方式。）

* 应从用户的角度来看什么任务更常用、什么任务更简单
* 并不一定非要进行调查获广泛研究
* 软盘格式化实用程序举例
* 没有考虑最简单、最常用和最核心的任务究竟是什么
* 让必要的确认和次要功能干扰了本来很简单的常用任务的顺利执行

1. 可见性原理（让完成给定用户所需的所有选项和材料对用户可见，不要让额外或冗余的信息干扰用户。）

* 在“所见即所得”的基础上，进一步实现“所见即所需”
* 设计目标是让所有需要和相关的选项可见和明确
* 不应让用户被不必要的信息所迷惑
* 传真机管理程序举例
* 功能的取名以及在用户界面上的位置安排存在问题
* 是一个由内向外的设计

1. 反馈原理（通过用户所熟悉的清楚、简洁和无歧义的语言，让用户时刻了解系统对用户操作的反应和解释，了解与用户有关且被他们所关心的系统状态变化、出错、异常等所有情况。）
   * + 成功的反馈就是以能被对方注意到、读到和正确理解的方式来提供信息

* 屏幕中央和顶端出现的反馈容易被注意
* 屏幕底端是最不容易被注意到的
  + - 好的出错信息
* 以一个能让用户马上知道问题何在的题头开始
* 有针对性和简洁地解释问题究竟是什么
* 建议解决问题的方法或操作步骤

1. 宽容原理（保持灵活和宽容，提供通过撤销和重做功能来减少用户出错和不当操作所带来的开销，同时保持允许各种不同的输入形式和顺序以及通过合理地解释用户的所有合理操作来尽可能防止出错。）
   * + 高可用性的系统帮助用户少犯错误
     + 宽容性如何取决于将什么数据检测为有效以及何时进行这种合法性检查

* 检查后，在返回给用户的屏幕上将所有不合法的数据域置为高亮度，把光标放在第一个出错的数据域，并且在状态栏中给出解释的话
* 过分使用合法性检查会降低可用性

1. 重用原理（对内部和外部的组件和行为加以重用，有目的而不是无目的地维持一致性，从而减少用户重新思考和记忆的需要。）

* 用户界面内在外观、位置以及行为上的一致性使得软件容易学习和记忆如何使用
* 界面更具有可预测性和更容易理解
* 重用程度越高，一致性就越好
* 不一致的用户界面不仅会降低软件的可用性，而且会增加程序设计的工作量

# Chapter9模型和理论

**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，特别适合于无法进行用户测试的情形，如：GOMS模型

## 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
* GOMS全称
* 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
* Operator-操作
* 任务执行的底层行为，不能分解
* 为达到目标而使用的认知过程和物理行为：如点击鼠标
* 操作时间是上下文无关的
* Method-方法
* 如何完成目标的过程，即对应目标的子目标序列和所需操作
* 如移动鼠标，输入关键字，点击Go按钮
* Selection-选择规则
* 确定当有多种方法时选择和方法
* GOMS认为方法的选择不是随机的
* 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
* 顺序关系
* 选择关系：以select：引导
* GOMS模型分析：
* 优点
* 能够容易地对不同的界面或系统进行比较分析
* GOMS有助于确定新产品的有效性
* 局限性
* 假设用户完全按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（顺序，选择）
* 忽略了用户间的个体差异

举例

使用GOMS模型描述在Word中删除文本的过程

 目标：删除Word中的文本

 方法1：使用菜单删除文本

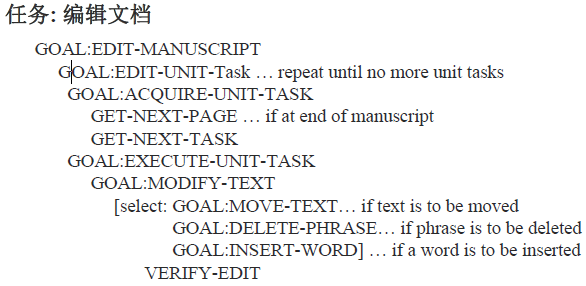
 步骤1：思考，需要选定待删除的文本

 步骤2：思考，应使用“剪裁”命令

 步骤3：思考，“剪裁”命令在“编辑”菜单中

 步骤4：选定待删除文本，执行“剪裁”命令

 步骤5：达到目标，返回



## 击键层次模型（KLM）

* 对用户执行情况进行量化预测
* 仅涉及任务性能的一个方面：时间
* 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
* 分析：
* 建模可以给出执行标准任务的时间
* 但没有考虑下面的问题：错误，学习性，功能性，回忆，专注程度，疲劳，可接受性

## Fitts定律

* 能够预测使用某种定位设备指向某个目标的时间
* 人机交互中，根据目标大小及至目标的距离，计算指向该目标的时间
* 可指导设计人员设计按钮的位置、大小和密集程度
* 对图形用户界面设计有明显的意义
* “最健壮并被广泛采用的人类运动模型之一”
* 定律内容：如果一个任务的困难程度可等价于“信息”，那么用户完成任务的速率即可等价于人类信息处理系统的“信息量”
* 三个部分：
* 困难指数ID (Index of Difficulty) = log2(*A*/*W*+1) (bits)
* 对任务困难程度的量化
* 与宽度和距离有关
* 运动时间MT (Movement Time) = a + b\*ID (secs)
* 在ID基础上将完成任务的时间量化
* 性能指数IP (Index of Performance) = ID/MT (bits/sec)
* 基于MT和ID的关系
* 也称吞吐量
* 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

# Chapter11 评估概论

DECIDE框架,时间；**任务的详细描述**

观察；询问；用户测试（重点）

注意方法的配对组合。不要全上，目标也是。

**交互评估框架：**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

## 基本概念

**评估原则**

* 评估应该依赖于产品的用户

与专业技术人员的水平和技术无关

* 评估与设计应结合进行

仅靠用户最后对产品的一两次评估，是不能全面反映出软件可用性的

* 评估应在用户的实际工作任务和操作环境下进行

根据用户完成任务的结果，进行客观的分析和评估

* 要选择有广泛代表性的用户
* 参加测试的人必须具有代表性

## 评估范型：

* 快速评估
* 设计人员非正式地向用户或顾问了解反馈信息，以证实设计构思是否符合用户需要
  + - 可在任何阶段进行
    - 强调“快速了解”，而非仔细记录研究发现
  + 如在设计初期了解用户对新产品的意见、在设计末期了解用户对图标设计的看法等
* 得到的数据通常是非正式、叙述性的
  + 可以口语、书面笔记、草图、场景的形式反馈到设计过程
* 是设计网站时常用的方法
* 基本特征：快速
* 可用性测试
* 20世纪80年代的主导方法
* 评测典型用户执行典型任务时的情况
* 包括用户出错次数、完成任务的时间等
* 基本特征
* 是在评估人员的密切控制之下实行的
* 主要任务
* 量化表示用户的执行情况
* 缺点
* 测试用户的数量通常较少
* 不适合进行细致的统计分析
* 实地研究
* 基本特征
* 在自然工作环境中进行
* 目的
* 理解用户的实际工作情形以及技术对他们的影响
* 作用
* 探索新技术的应用契机
* 确定产品的需求
* 促进技术的引入
* 评估技术的应用
* 分类
* 评测人员作为“局外人”
* 评测人员也可作为“局内人”或测试用户
* 预测性评估
* 专家们根据自己对典型用户的了解（通常使用启发式过程）预测可用性问题
* 也可使用理论模型
* 基本特征
* 用户可以不在场
* 使得整个过程快速、成本较低
* 启发性评估是典型的预测性评估方法
* 注意：启发式原则应定制
* 且有些结果可能并不准确

## 评估技术

1. 观察用户

* 有助于确定新产品的需求
* 也可用于评估原型
* 挑战：如何在不干扰用户的前提下观察用户，以及如何分析大量数据

1. 询问用户意见

* 简单，调查用户数量从几个到几百不等

1. 询问专家意见

* “角色扮演”方式评估
* 同时专家会提出解决方案

1. 测试用户的执行情况

* 可比较不同设计方案优劣
* 通常在受控环境中进行

1. 基于模型和理论，预测界面的有效性

* 常用技术如GOMS模型和KLM模型等

**评估技术比较（这个图很重要！）**



## 评估步骤

**（DECIDE框架用于指导评估过程）：**

1. D决定评估需要完成的总体目标

评估目标决定了评估过程，影响评估范型的选择。产品设计是否了解用户需要？界面是否满足一致性要求？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

1. E发掘需要回答的具体问题

根据目标确定问题

1. C选择用于回答具体问题的评估范型和技术

范型决定技术类型，需要权衡实际问题和道德问题，可结合使用多种技术

1. I标识必须解决的实际问题，如测试用户的选择，设施和设备的准备、预算和培训等
2. D决定如何处理有关道德的问题

应保护个人隐私（可在评估前签署一份协议书），明确测试对象是软件，而非个人。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

1. E评估、解释并表示数据

决定应搜集什么数据，如何分析，如何表示数据

可用性问题——评估结果总是可用性问题清单以及改进建议

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

# Chapter12 评估之观察用户

## 观察方式

1. 真实环境中的观察

* 观察者既可作为旁观者，也可作为参与者
* 重点是应用的上下文

1. 受控环境中的观察

* 观察者不能作为参与者
* 重点是研究用户执行任务的细节

1. 二者差别不大

* 有时前者模仿后者的测试条件
* 实地观察也可作为实验室观察的补充

**实验室观察：**

* 在专门为可用性测试而安装配置的固定设备的环境下进行的观察
* 优点
* 提供了可控且一致的评估环境，易于分析比较
* 缺点
* 可控且一致的评估环境：人为环境、不自然，可能降低测试结论的普遍性和一般性
* 不利于观察多人之间的协作
* 具备观察测试过程的能力很重要
* 对开发人员
* 令开发人员感到丢脸面
* 给他们一些启发和指导
* 使之能够设身处地地为最终用户着想
* 文档准备
* 协议书
* 要求用户签署
* 说明测试的目的、测试时间，并解释他们的权利
* 让用户感觉轻松自在
* 脚本：要求用户执行的任务

**现场观察：**

* 指在用户的实际环境中观察用户在使用软件时的情况
* 是发现同使用环境有关的问题的最佳手段

## 数据记录

可以根据研究人员的专业素质及环境、项目的特点来选取合适的方法

1. 纸笔记录

* 最原始、最廉价
* 前提：对观察对象有一定的了解，从而有明确的观察侧重点
* 优点：事后对观察结果进行分析的工作量小
* 缺点：
  + 观察者容易疲劳，而且记录速度有限
  + 建议将记录者和评估者分开

1. 音视频记录

* 适用场合
* 对于观察对象不太了解，或者是需要观察的内容较多
* 特别是在采用边做边说法时
* 音频记录
* 信息全面，没有任何遗漏，便于事后详细分析
* 没有可见记录，转录数据非常烦琐
* 常用于提示重要细节或作为情景说明的辅助材料
* 视频记录
* 能够看到参与者正在做的事情
* 但要始终让参与者停留在视觉的范围内很困难
* 缺点
* 所含的信息量很大，所以数据分析非常耗时

1. 日志和交互记录

* 间接观察的适用场合
* 直接观察可能影响用户
* 或者评估人员无法在现场进行研究
* 可根据搜集到的数据，推断实际情形，并找出可用性和用户体验方面的问题
* 优点
* 体现了用户是如何完成真实任务的
* 使得从工作在不同环境下的大量用户那里自动收集数据变得相当容易
* 适用于用户分散、无法当面测试的情形
* 如互联网应用和网站设计项目等

## 数据分析

* 数据类型

1. 用于描述的定性数据：描述观察到的现象
2. 用于分类的定性数据：使用各种技术进行分类
3. 定量数据：用于统计目的

* 定性分析：
* 分析方法：有时候详细分析不必要，粗略分析即可，结合上下文研究具体动作
* 常用方法：找出关键事件，如用户遇到困难的地方
* 内容分析（content analysis）
* 用于“详细分析”录像数据
* 把数据内容划分为一些有意义、而且互斥的类别
* 不能以任何方式相互重叠、内容类别也必须可靠
* 可靠：不同人的分类结果不能存在很大差异
* 费时、费力、不常使用
* 改进：层次化内容分类技术
* 会话分析（conversation analysis）
* 用于仔细检查语义，重点是对话过程
* 弄明白发话者想要表达什么，而受话者又是怎么样理解它的意思并做出反应
* 可用于聊天室、虚拟社区等互联网应用，以增进对用户需要的理解
* 话语分析（discourse analysis）
* 关注话语的使用，而不是内容
* 把语言视为反映心理和社会因素的媒介
* 了解人们如何使用语言
* 措辞上的微小改动即可改变话语的意思
* 当你说“我正在进行话语分析”时，你实际上就是在进行话语分析……
* 张三认为，当你说“我正在进行话语分析”时，你实际上就是在进行话语分析……
* 定量分析：
* 采用录像记录时，当发现错误或异常操作时，观察的同时要对数据做标记和简短说明
* 统计分析

# Chapter13 评估之询问用户和专家

## 询问用户之访谈

* 指导原则
* 避免过长的问题
* 避免使用复合句
* 避免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
* 访谈步骤
* “开始”阶段
  + 访问人先介绍自己
  + 解释访谈的原因，消除受访人对道德问题的疑虑，询问受访人是否介意被记录（录音或摄像）
* “热身”阶段
* 先提出简单的问题
* 主要访谈阶段
* 按逻辑次序由易到难提问
* “冷却”阶段
* 提出若干容易的问题，消除用户的紧张感觉
* 结束访谈
* 感谢受访者，关闭录音机，收好笔记本，表面访谈已经结束
* 访谈类型
* 非结构化访谈
* 问题是开放式的，不限定内容和格式
* 受访人自行选择详细回答还是简要回答
* 访问人应确保能够搜集到重要问题的回答
* 结构化访谈
* 根据预先确定的一组问题进行访谈
* 问题通常是“封闭式”的，它要求准确的回答
* 半结构化访谈
* 开放式问题+封闭式问题
* 注意不要暗示答案
* 集体访谈
* 基本思想：个别成员的看法是在应用的上下文中通过与其他用户的交流而形成的
* “焦点小组”是集体访谈的一种形式

## 询问用户之问卷调查（重要）

* 问卷调查是用于搜集统计数据和用户意见的常用方法
* 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
* 做到直观、一致
* 避免使用术语
* 明确说明如何完成问卷
* 如说明应在选项前的方框内打“√”
* 在设计问卷时，既要做到紧凑，也应适当留空
* 问题类型
* 常规问题
* 年龄、性别、职业、居住地、应用计算机的经验等
* 自由回答问题
* 如：你能够对这个界面提出改进意见吗？
* 能够提出设计人员没有考虑到的建议
* 量化分级问题
* 要求用户以数值尺度判断一个特定陈述
* 如：系统容易从错误状态恢复
* 不同意1 2 3 4 5 同意
* 第三章中的“Likert尺度”和“语义差异度尺度”
* 奇数刻度较偶数刻度更常用
* 多选题
* 对于收集用户以前的经验信息很有用
* 在线问卷调查

两种形式：

* 基于电子邮件：能够针对特定的用户，但邮件能够容纳的内容有限
* 基于网页的调查：形式灵活，并能验证数据的有效性，但调查对象是随机的
* 问卷调查与访谈
* 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
* 询问ZAP命令的说明
* 系统新增功能的问卷
* 移动电话说明书的问卷
* 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

## 询问专家之认知走查

* 逐步检查使用系统执行任务的过程，从中找出可用性问题
* 无需用户参与
* 认知走查的主要目标是确定使一个系统如何易于学习
* 试图想象出人们在第一次使用某个产品时的想法以及所采取的动作，它的大作流程是怎样的
* 评估的具体过程就是把用户在完成这个功能时所做的所有动作讲述成一个令人可以信服的故事
* 走查的步骤：
  + - 标识并记录典型用户的特性
    - 基于评估重点，设计样本任务
    - 制作界面原型（或界面描述），明确用户执行任务的具体步骤
    - 由设计人员和专家级评估人员（一位或多位）共同进行分析
    - 评估人员结合应用的上下文，逐步检查每项任务的操作步骤
    - 在完成逐步检查之后，汇总关键信息
    - 修改设计，更正发现的问题
* 认知走查的记录工作非常重要！
* 分析：
* 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
* 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
* 协作走查
  + - 由用户、开发人员和可用性专家合作，逐步检查任务场景，讨论与对话元素相关的可用性问题
* 在评估过程中，每一位专家都承担用户的角色
  + - 优点
* 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
* 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

## 询问专家之启发式评估

* 一种灵活而又相当廉价的评估方式
* 评估步骤（如何执行）
  + - 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
* 至少应检查界面两遍
  + - 第一遍主要了解产品的范围和交互过程
    - 第二遍则专注于具体的界面元素，并结合应用的上下文，找出潜在的可用性问题
* 记录潜在可用性问题和严重程度
  + - 总结阶段
* 专家们集中讨论评估发现，确定对界面改进或重新设计方面的建议
* 优点：
  + - 由于不涉及用户，所以面临的实际限制和道德问题较少
    - 成本相对较低，不需要特殊设备，而且较为快捷
    - 又被称为“经济评估法”
* 缺点
* 评估人员需要经过长时间的训练才能成为专家
* 理想的专家应同时具备交互设计和产品应用域的知识
* 可能出现“虚假警报”
* “专家每找到一个真实的可用性问题，将发出约一个假警报（1.2），忽略大约半个问题（0.6）”
* 七项启发式原则

1. 内部一致性：用户能明确所有措辞的含义
2. 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
3. 快捷链接：适合无经验和有经验用户使用
4. 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
5. 预防错误：能够预防错误的发生
6. 提示信息：应能掌握系统的运行状态
7. 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

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

# Chapter14评估之用户测试

**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

**科学实验：**

* 目的是发现新知识
* 如为了决定需要在鼠标上设计多少个按键而进行的实验
* 如何设计图标等

**测试设计**

* 用户测试须考虑实际限制并做出适当的折衷
  + - 应确保不同参与者的测试条件相同
    - 应确保评估目标特征具有代表性
    - 实验可重复，但通常不能得到完全相同的结果
    - 以DECIDE框架为基础
* 1：定义目标和问题
  + - 目标描述了开展一个测试的原因，定义了测试在整个项目中的价值
    - 问题是对关注点的说明和解答
* 2：选择参与者



* 3：设计测试任务
  + - 测试任务应当与定义的目标相关
    - 测试任务通常是简单任务，如查找信息
    - 有时采用较为复杂的任务，如加入在线社团等
    - 任务不能仅限于所要测试的功能，应使用户全面的使用设计的各个区域
* 如关注搜索功能的可用性，可请求参与者搜索找出产品X
* 更好的方法就是请求参与者找出产品X并同产品Y进行比较
* 每项任务的时间应介于5~20分钟
* 应当以某些合乎逻辑的方法安排任务
* 开始时，先提出简单问题有助于增强用户的自信心
* 4：明确测试步骤
* 在测试之前，准备好测试进度表和说明，设置好各种设备
* 正式测试前应进行小规模测试
* 在必要时，评估人员应询问参与者遇到了什么问题
* 若用户确实无法完成某些任务，应让他们继续下一项任务
* 测试过程应控制在1小时之内
* 必须分析所有搜集到的数据
* 5：数据搜集
* 确定如何度量观测的结果
* 使用的度量类型依赖于所选择的任务
* 定量度量和定性度量
* 6：数据分析
* 变量
  + - * 实验的目的是回答某个问题或测试某个假设，从而揭示两个或更多事件之间的关系
      * 这些“事件”称之为“变量”
* 自变量
* 为回答假设问题，需被操作的一个或多个变量，即开始实验之前，已经设置好的变量
* 复杂的实验可能包含不止一个自变量，如假设用户的反应速度不仅取决于菜单选项的数目，也取决于菜单中应用的命令选择
* 因变量
* 能在实验中测量的变量
* 其值依赖于自变量的变化
* 如：完成任务所花费的时间、出错的数目、用户偏爱和用户执行的质量
* 7：总结、报告测试结果

# Chapter10 用户为中心

**以产品为中心的设计有哪些问题？**

没有在把设计人员行动与用户关联起来方面提供很多东西，设计人员和用户之间的交互只限于需求和规格说明文档，只限于最后的交付。设计过程主要关注的是机器及其效率，期望人能够使用机器。

## 以用户为中心的设计（UCD）思想：

* 三个方面的假设
* 好的设计结果使客户感到满意
* 设计过程是设计人员与客户之间的协作过程
* 设计要进化并适应客户不断变化的考虑
* 规格说明是该过程的副产品
* 在整个过程中，客户和设计人员要不断沟通
* 四项重要原则
* 及早以用户为中心
* 合设计——设计的所有方面应当齐头并进地发展
* 及早并持续性地进行测试
* 迭代设计
* 包含的方法
* 用户参与（user participation）
* 焦点小组（focus groups）
* 问卷调查
* 民族志观察
* 走查
* 专家评估
* 可用性测试

## 参与式设计：

* 让用户参与开发的一种方法
* 用户积极参与开发
* 即成为设计小组成员，与设计人员合作设计产品
* 制作模型能够有效地利用用户的经验和知识：PICTIVE和CARD

PICTIVE

* 协作式产品的界面造型技术
* 使用低保真的办公室用品模型来研究系统的特定屏幕和窗口布局
* 目的
* 使得用户能够参与设计过程
* 改进设计过程的知识获取方法
* 四个阶段
* 当事人做自我介绍
* 简短讲解说明不同应用域
* 围绕设计的集策讨论
* 设计走查和决策讨论

CARD

* 需求和设计的协作分析
* 使用画有计算机和屏幕图像的卡片发掘各种工作流
* 是“情节串联图”的一种形式

CARD vs. PICTIVE

* 侧重点不同：PICTIVE关注的是系统细节，而CARD注重的是宏观的任务流
* 可互为补充

## 理解用户工作：

1. 观察法
2. 上下文询问法：

* 观察并与用户交流会比仅仅观察的效果要好
* 强调到用户工作的地方，在用户工作时观察，并和用户讨论他的工作
* 也被翻译成“情景调查”
* 基于“学徒模型”

用户是师傅，研究人员是学徒

* 与观察法的区别
* 用户知道研究人员的存在
* 也知道他们是研究的一部分
* 4个原则
* 上下文环境
  + 应深入工作空间，以了解其中发生的事情
  + 可以要求用户发声思考，也可以只在必要时发问
* 伙伴关系
* 开发人员和用户应相互合作
* 提醒用户是专家，将研究人员作为新手
* 解释
* 解释过程必须由用户和开发人员合作完成
* 杜绝设计人员片面地对事实作出解释或假设
* 焦点
* 把问题集中在所定的研究题目上
* 准备一个观察方向的列表
* 上下文询问法vs.民族志观察
  + - 过程更简短
* 2至3个小时vs. 数周或数月
  + - 重点更为明确、集中
* 民族志观察的角度更广
* 设计人员只询问，不参与
  + - 目的明确—设计新系统

## 以用户为中心的设计方法的反思？

**以用户为中心方法的缺陷**

 影响产品的创新性；

 可操作性受到时间、预算和任务规模的限制；

 忽视了人的主观能动性和对技术的适应能力；

**以活动为中心的设计思想**

 把用户要做的“事”（活动）作为重点关注的对象

 更适合于复杂的设计项目

**ACD思想是对UCD思想的一种反思**

 ACD把人与技术综合起来进行考虑，不单纯考虑人或者技术，而是关注事情本身的活动目标

 ACD同样需要对用户进行研究或调研

**倾听用户永远是明智的，但屈从于用户的要求是不明智的**