

# 用户界面设计

---

- 为软件系统设计有效的界面

# 目标

---

- 用户界面设计的一般设计原则
- 软件系统的几种不同的交互方式
- 不同的信息表示方式
- 软件内建的用户支持的设计基础
- 可用性属性以及系统评价的简单方法

# 内容

---

- 用户界面设计原则
- 用户交互
- 信息表示
- 用户支持
- 界面评价

# 用户界面

---

- 系统用户经常通过界面来判断系统，而不是功能。
- 设计得不好的界面会导致用户犯下灾难性的错误
- 不好的用户界面是导致许多软件系统不被使用的原因。

# 图形化用户界面

---

- 大多数商业系统的用户通过图形化界面和系统交互，虽然有些情况下，传统的基于文本的界面还在使用。

# GUI的特点

---

特性	描述
窗口	多窗口允许不同的信息被同时显示在用户屏幕上
图标	图标代表不同类型的信息。在一些系统中，图标代表文件，而在另外的地方，图标代表过程。
菜单	命令是通过菜单选择的而不是通过键入字符命令
鼠标	通过鼠标从菜单中选择或者指点窗口中的内容
图形	在同一个显示中可以既有图形也有文字

# GUI 的优点

---

- 便于学习和使用。
  - 没有经验的用户可以快速的学会使用系统
- 用户可以快速地从一個任务切换到另一个任务，可以同时跟几个不同的应用程序交互
  - 当窗口切换时，信息仍在各自的窗口中保持可见
- 可以快速的，全屏幕的交互

# 用户为中心的设计

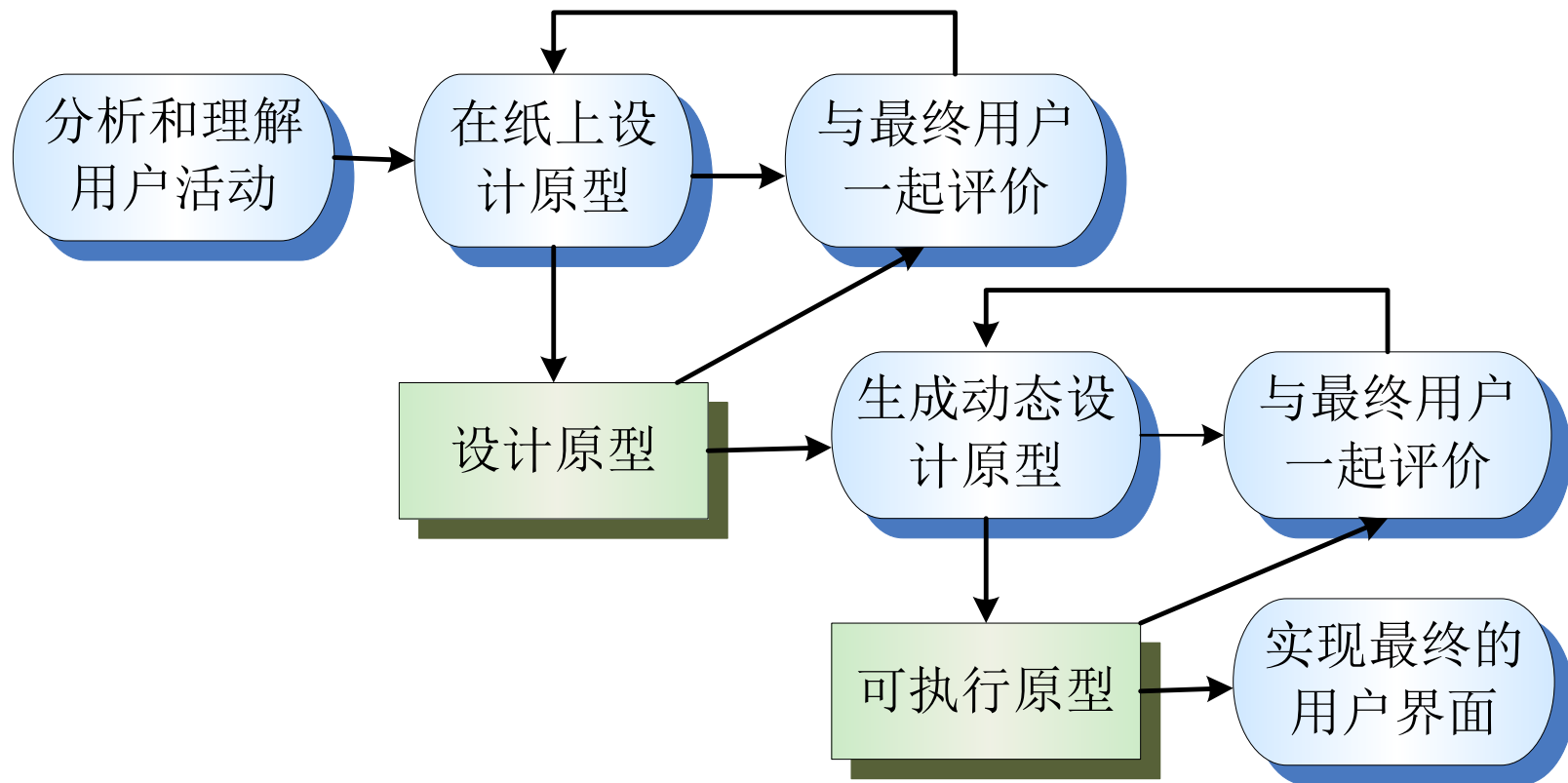
---

- 本章的目的是介绍用户界面的设计，而不是实现。
- 以用户为中心的界面设计方法中，用户的需要是最为重要的，用户也参与到设计过程中来。
- 用户界面设计牵涉到原型界面的开发。



# 用户界面设计过程

---



# 用户界面设计原则

---

- 用户界面设计必须考虑系统使用者的需要、经验和能力
- 设计者必须意识到人们身体上和智力上的限制（如短时间记忆上的限制），必须意识到人们会犯错误。
- 不是所有的用户界面设计原则都适用于所有设计

# 用户界面设计的原则

---

原则	描述
用户熟悉	界面所应用的术语和概念应该是来自于用户的经验，这些用户是将要使用系统最多的人
一致性	界面应该是一致的，即尽可能地让相似的操作有同样的触发方式
意外最小化	永远不要让用户对系统的行为感到吃惊
可恢复性	界面应该有一种机制来允许用户从错误中恢复
用户指南	在错误发生时界面应该提供有意义的反馈，并有上下文感知能力的用户帮助功能
用户差异性	界面应该为不同类型用户提供合适的交互功能

# 设计原则

---

- 用户熟悉
  - 界面应该使用面向用户的术语和概念而不是计算机的概念。比如，Office系统应该使用信函、文档、文件夹这样的术语，而不是目录、文件识别号等术语。
- 一致性
  - 系统界面应该具有一定程度的一致性。命令和菜单应该具有相同的格式，符号应该类似。
- 意外最小化
  - 类似的操作用户会有类似的预期

# 设计原则

---

- 可恢复性
  - 系统应该提供允许用户从错误中恢复。比如Undo功能，破坏性操作前的确认动作等。
- 用户辅助
  - 需要提供用户辅助功能，例如帮助系统，在线手册等。
- 用户差异性
  - 要支持不同类型用户的交互功能。比如，有些用户视力不好，需要有更大号的字体。

# 用户—系统交互

---

- 在交互系统设计中要考虑两个问题
  - 用户的信息如何提供给计算机系统?
  - 计算机系统的信息如何显示给用户看?
- 用户交互和信息表示可以通过一个连贯的平台集成在一起

# 交互形式

---

- 直接操作
- 菜单选择
- 表格填写
- 命令语言
- 自然语言

## 各种交互形式的优点和缺点

交互类型	主要优点	主要缺点	应用实例
直接操作	快速和直观的交互 容易学习	较难实现 只适合于任务和对象有视觉 隐喻的情况	视频游戏 CAD系统
菜单选择	避免用户错误 只需很少的键盘输入	对有经验用户操作较慢 当菜单选择很多时会变得很 复杂	绝大多数一般用途的系统
表格填写	简单的数据入口 容易学习	占据很多屏幕空间	库存控制 个人贷款处理
命令语言	强大灵活	较难学习 差的错误管理	操作系统 图书馆信息检索系统
自然语言	适合偶然用户 容易扩展	需要键入的太多 自然语言理解系统不可靠	时刻表系统 WWW信息检索系统



# 直接操作的优点

---

- 用户感觉到是在控制计算机而不是受制于计算机
- 用户的学习实践较短
- 用户操作的时候能立即得到反馈，所以操作错误能够很快察觉并得到纠正。

# 直接操作的问题

---

- 建立一个合适的信息空间模型比较困难
- 对于一个很大的信息空间，怎样提供合适的导航工具？
- 直接操作的界面对程序来说比较复杂，对计算机系统来说负担也较重

# 控制面板界面

---

Title	JSD. example	<input type="checkbox"/>	Grid	Busy
Method	JSD			
Type	Network	Units	cm ▶	QUIT
Selection	Process	Reduce	Full ▶	PRINT
NODE   LINKS   FONT   LABEL   EDIT				

# 菜单系统

---

- 用户从一个系统可能的动作一览中作出选择
- 选择可以用鼠标的移动和点击、移动方向键或者输入选择项的名称
- 使用易用的终端，如触摸屏

# 菜单系统的优点

---

- 用户不需记忆命令
- 输入操作很少
- 用户错误可以由界面捕获
- 可以提供基于上下文的在线帮助。用户的上下文由基于当前菜单选择。

# 菜单系统的问题

---

- 逻辑上关联的动作难于表达
- 菜单系统适合于数量不大的命令选择。数量大的时候需要有菜单构建工具
- 有经验的用户会觉得菜单比命令语言慢

# 基于表格的界面

---

NEW BOOK			
Title	<input type="text"/>	ISBN	<input type="text"/>
Author	<input type="text"/>	Price	<input type="text"/>
Publisher	<input type="text"/>	Publication date	<input type="text"/>
Edition	<input type="text"/>	Number of copies	<input type="text"/>
Classification	<input type="text"/>	Loan status	<input type="text"/>
Date of purchase	<input type="text"/>	Order status	<input type="text"/>

# 命令界面

---

- 用户输入命令以指示系统动作，例如，UNIX
- 可以用较为低廉的终端实现。
- 用编译技术容易处理
- 复杂的命令可以通过命令合并产生
- 可以生成一个只需较少输入的简明的界面



# 命令界面的问题

---

- 用户必须记住命令，所以命令语言不适合临时性的用户
- 用户容易输错命令，所以需要有一个错误探测和恢复系统
- 系统交互通过键盘进行，所以需要有一定的打字输入能力

# 命令语言

---

- 有经验的用户比较喜欢，因为这样他们可以较快地和系统交互
- 不适合于临时性的、以及没经验的用户
- 可以作为菜单命令的一个替补 (快捷键)。有些情况下，命令方式和菜单方式同时提供。

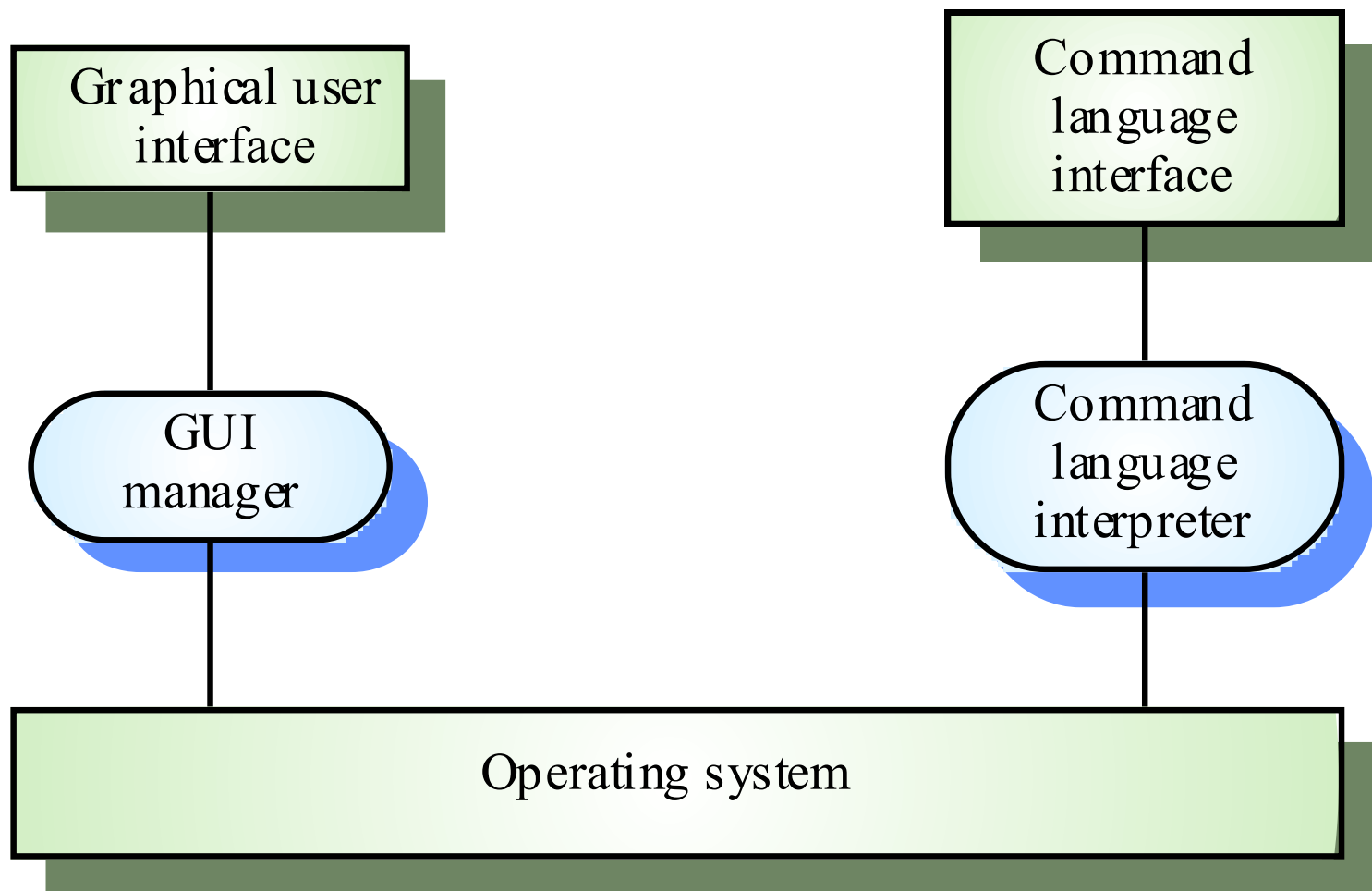
# 自然语言界面

---

- 用户输入自然语言的命令。通常词汇是限定的，这样的系统也只限于一些特定的应用领域 (e.g. 时刻表查询)
- 自然语言处理技术还不成熟，对临时性的用户来说还不是很有效，而有经验的用户又会觉得输入量太多

# 多重用户界面

---



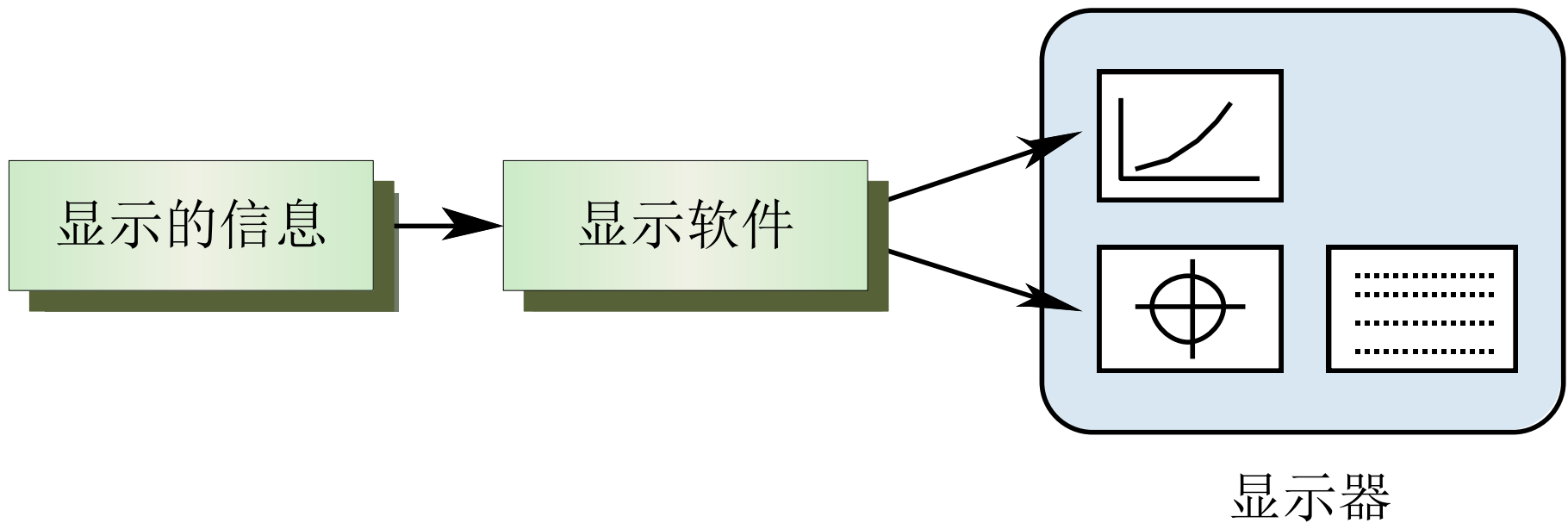
# 信息表示

---

- 将系统的信息显示给系统用户
- 信息可以直接显示 (e.g. 字处理软件中的文本) , 或者可以转换成某种方式来显示 (e.g. 图形化的方式)
- MVC (Model-View-Controller) 方法支持数据的多种表示

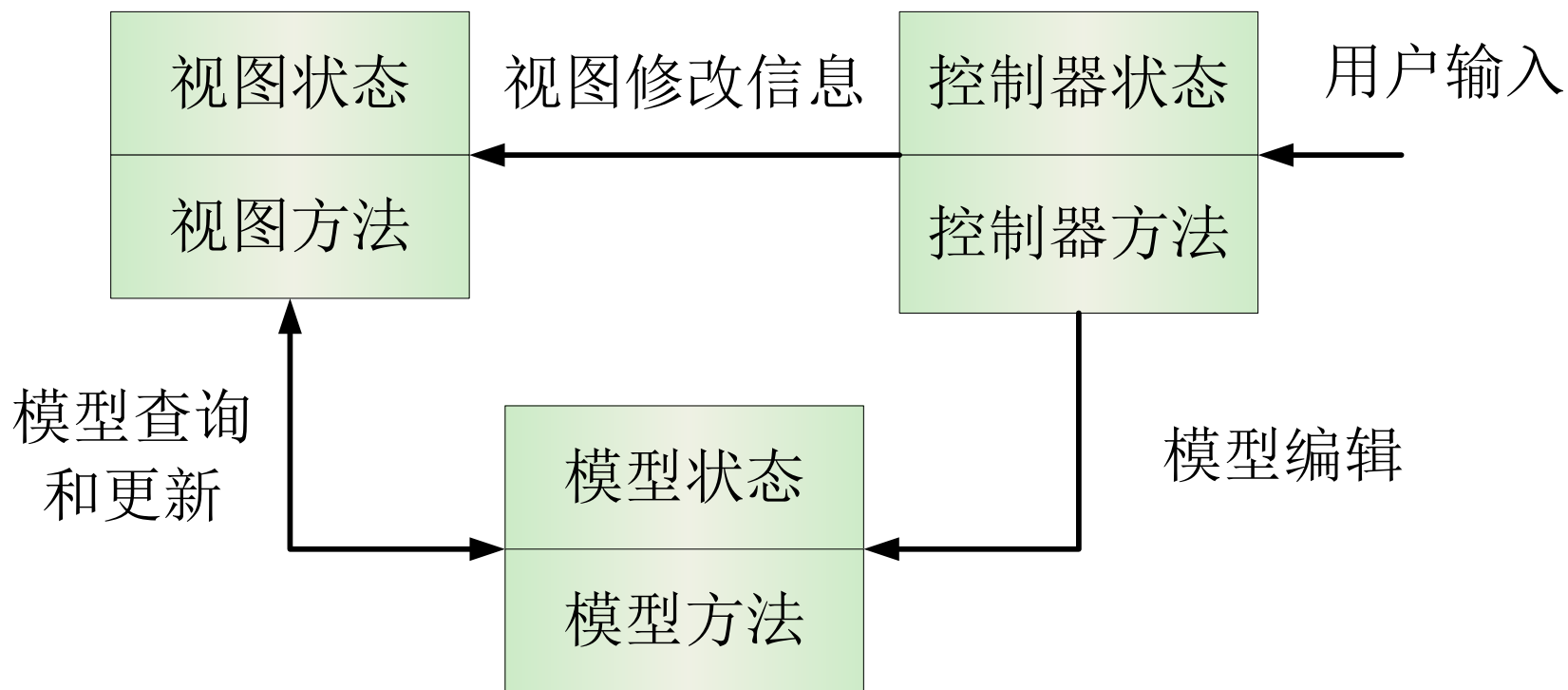
# 信息表示

---



# MVC方法

---



# 信息表示

---

- 静态信息
  - 一次会话的开始时初始化，在会话过程中不改变。
  - 可以是数字或者文字
- 动态信息
  - 在会话过程中改变，改变的信息传达到系统用户
  - 可以是数字或者文字



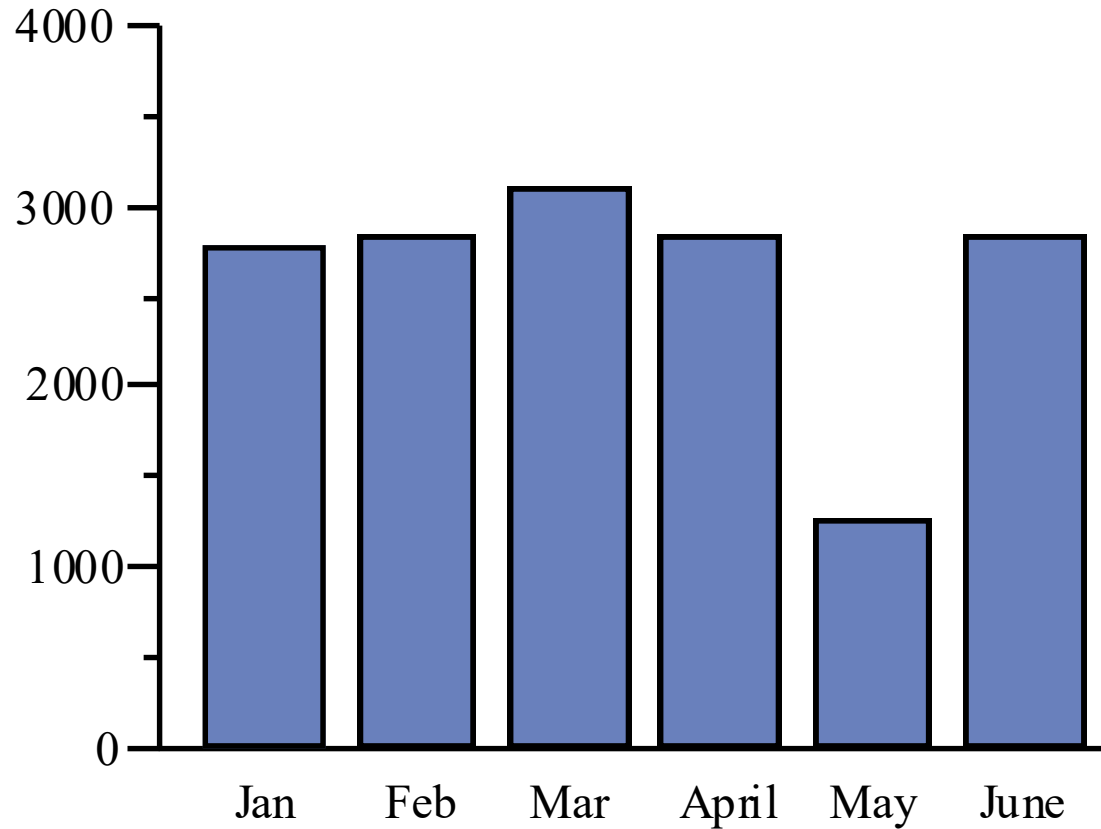
# 信息显示的要點

---

- 用户对精确信息或不同数据值之间的关系感兴趣吗?
- 信息值变化的速度如何?数值的变化需要马上显示给用户吗?
- 用户必须响应信息的变化执行某种动作吗?
- 需要使用直接操作界面吗?
- 信息是文本还是数字? 信息的相对值是否重要?

# 两种信息表示方法

Jan	Feb	Mar	April	May	June
2842	2851	3164	2789	1273	2835



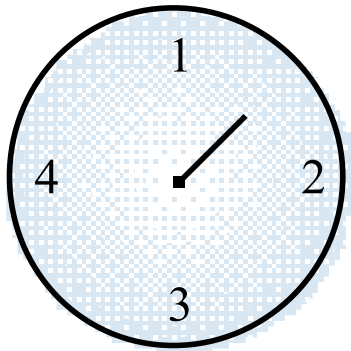
# 模拟的vs.数字的表示方法

---

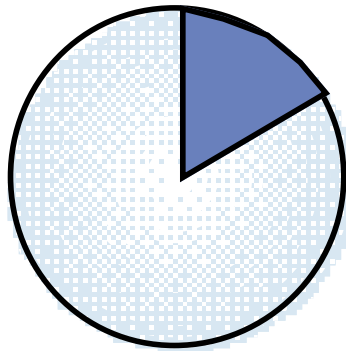
- 数字表示
  - 紧凑 – 需要较少的屏幕空间
  - 可以传达精确值
- 模拟表示
  - “扫一眼”就能得到一个值的印象
  - 可以表示相对值
  - 容易显示异常值

# 动态信息显示

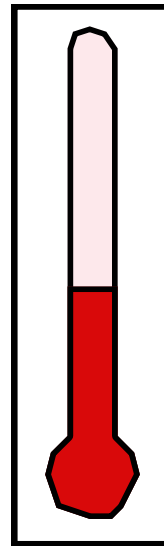
---



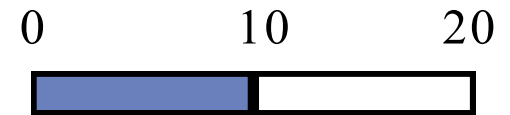
Dial with needle



Pie chart



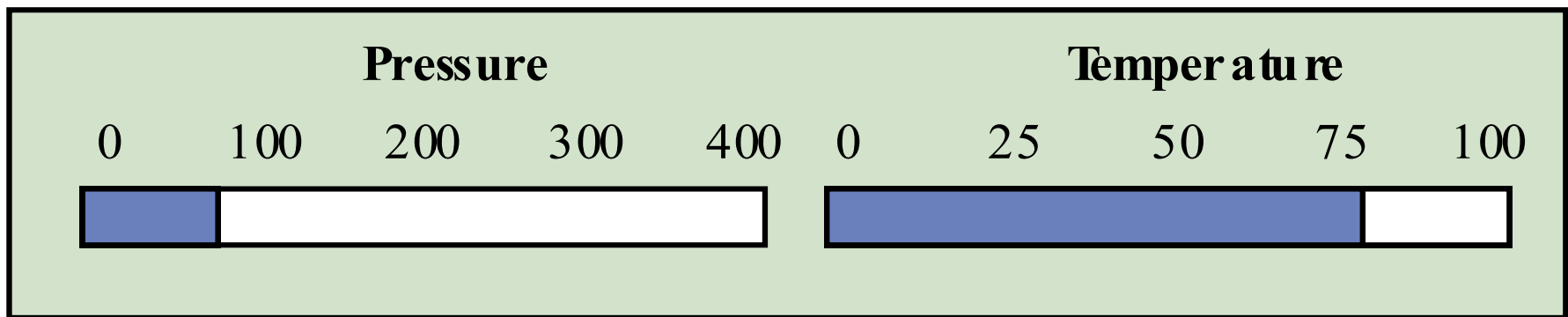
Thermometer



Horizontal bar

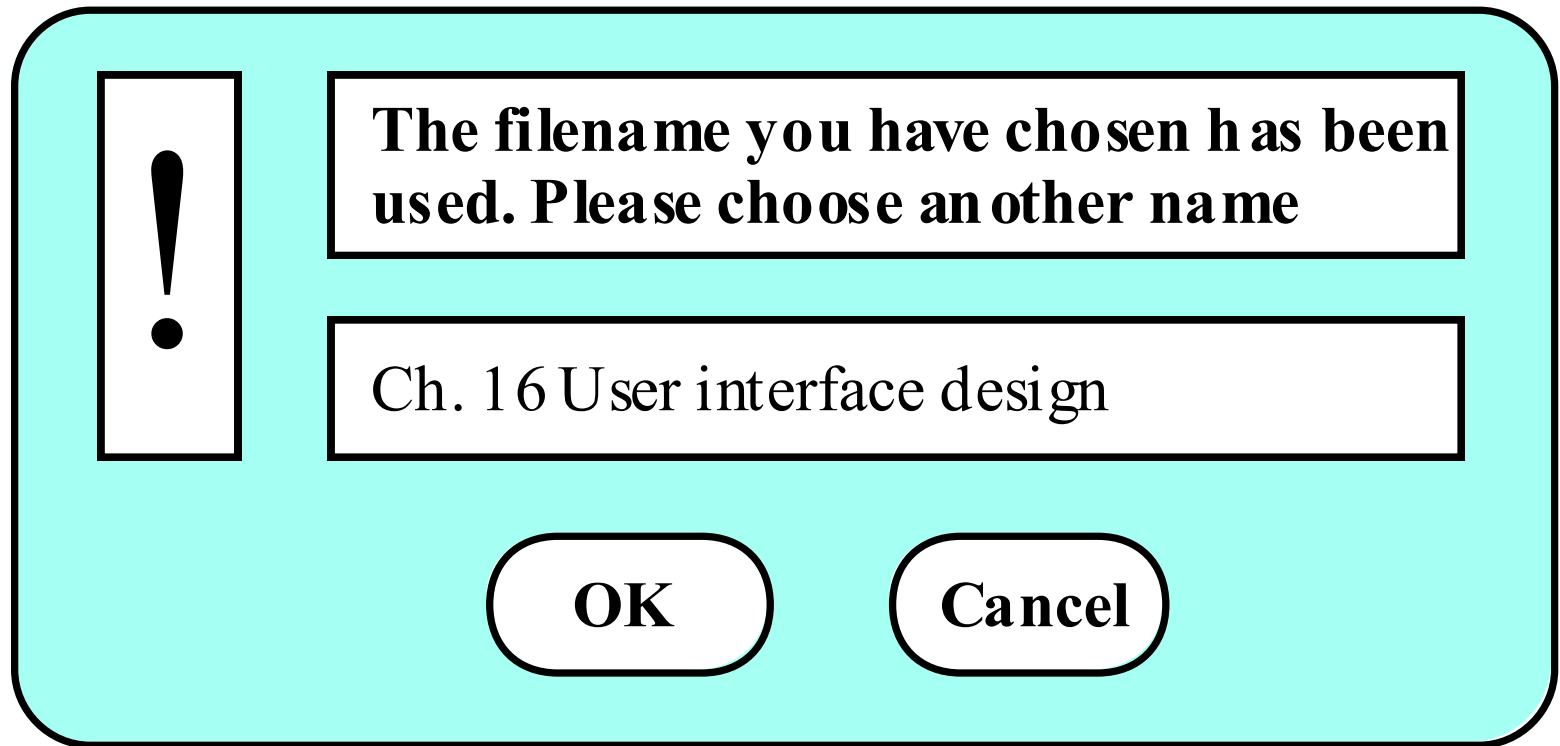
# 显示相对值

---



# 文本的突出显示

---



# 数据的可视化

---

- 显示大量信息的技术
- 可视化可以揭示出实体和数据趋势间的关系
- 可视化实例:
  - 从多个地点采集的气象信息
  - 电话网络状态表示成一组相互连接的节点
  - 化工厂可视化为的一组相互连接的反应罐和管道，显示压力和温度
  - 3维分子模型
  - 一组网页显示为一个扩展树

# 色彩显示

---

- 色彩给界面增加了额外的信息，可以帮助用户理解复杂的信息结构
- 可以用来突出显示异常事件
- 界面设计中色彩使用的常见问题：
  - 用颜色传达特定的含义
  - 过度使用颜色



# 使用色彩的指南

---

- 不要使用太多的颜色
- 使用不同颜色代表不同的任务
- 允许用户控制颜色编码
- 先用单色设计然后加上颜色
- 颜色编码要注意前后一致
- 避免互相冲突的颜色配对
- 用颜色变化来表示状态变化
- 要注意颜色的显示具有较低的分辨率

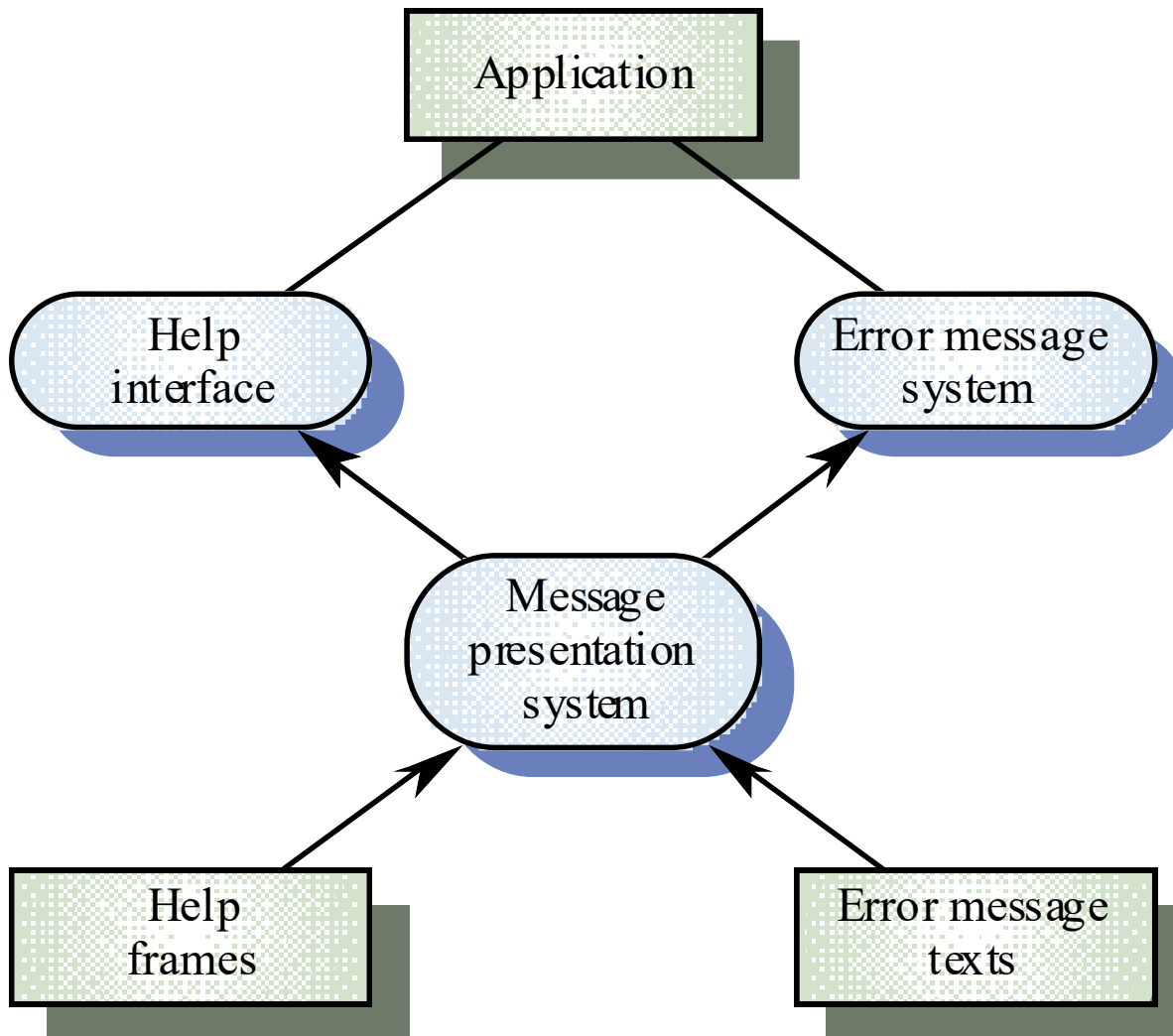
# 用户支持

---

- 用户指南包括：在线帮助,错误信息,手册等。
- 用户指南视同应该和用户界面集成在一起，这样当用户需要或者用户犯了错误的时候可以帮助用户
- 如果可能的话，帮助和消息系统应该集成在一起

# 帮助和消息系统

---



# 错误消息

---

- 错误消息的设计非常重要。  
不好的错误消息会使得用户拒绝使用系统。
- 消息应该有礼貌的、简明的、一致的、建设性的。
- 用户的背景和经验是消息设计的决定性因素。

# 消息措辞的设计因素

---

因素	描述
上下文	用户指南系统应该能注意到用户当前在干什么，针对当前上下文调整输出消息
经验	因为用户逐渐熟悉了系统，就会对冗长的、过于详细的消息不满。但初学者可能还嫌问题阐述得不够清楚。用户指南系统应该根据不同类型的用户对象选择不同的表达方式和所用的术语。
技能水平	消息应该根据用户的技能进行裁剪。消息应该根据不同类型的用户对象选择不同的表达方式和所用的术语。
风格	消息应该是积极的而不是消极的。应该以主动方式去表示出来而不是被动显示。决不能是无礼的或者是滑稽怪诞的。
文化	消息的设计应该尽可能熟悉所在国的文化传统。一条消息对一个地区是合适的，而在另一个地区可能不可接受。

# 护士输入病人姓名

---

请在方框内键入病人名字，并按确认键

病人姓名

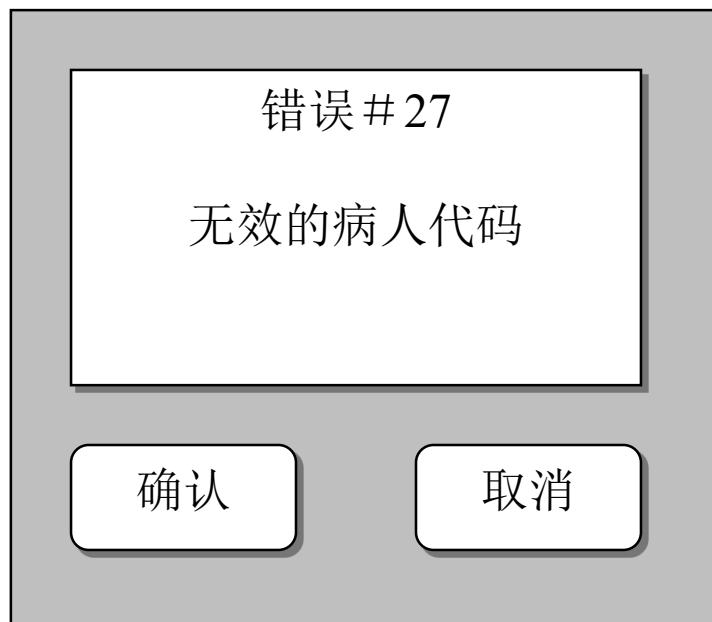
张三

确认

取消

# 面向系统和面向用户的错误消息

---

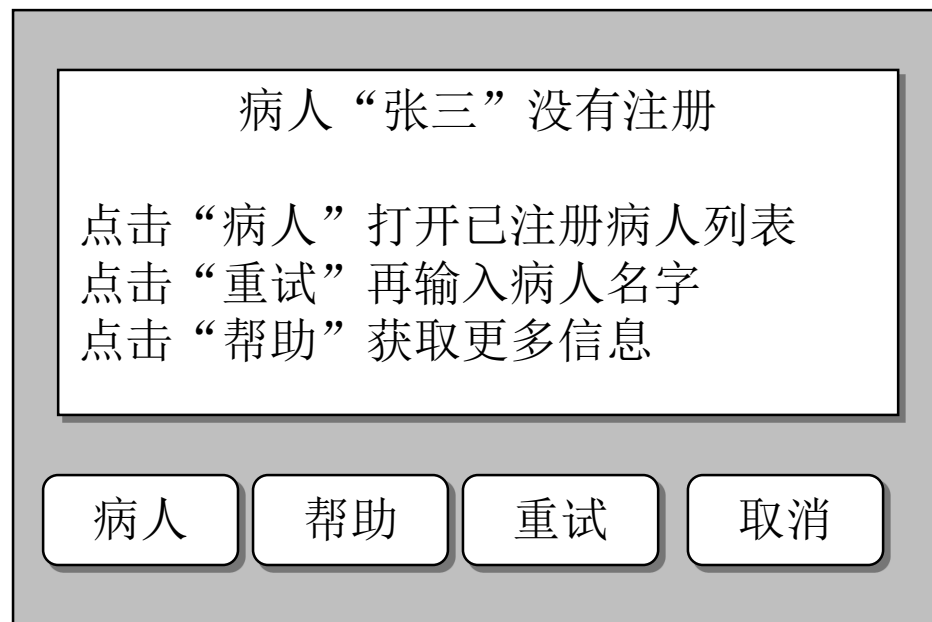


错误 #27

无效的病人代码

确认 取消

This is a system-oriented error message dialog box. It has a light gray background and a white message area. The message area contains the text "错误 #27" (Error #27) and "无效的病人代码" (Invalid patient code). Below the message area are two buttons: "确认" (Confirm) and "取消" (Cancel).



病人“张三”没有注册

点击“病人”打开已注册病人列表  
点击“重试”再输入病人名字  
点击“帮助”获取更多信息

病人 帮助 重试 取消

This is a user-oriented error message dialog box. It has a light gray background and a white message area. The message area contains the text "病人“张三”没有注册" (Patient "Zhang San" is not registered) and three lines of instructions: "点击“病人”打开已注册病人列表" (Click "Patient" to open the list of registered patients), "点击“重试”再输入病人名字" (Click "Retry" to re-enter the patient's name), and "点击“帮助”获取更多信息" (Click "Help" to get more information). Below the message area are four buttons: "病人" (Patient), "帮助" (Help), "重试" (Retry), and "取消" (Cancel).

# 帮助系统设计

---

- 第一种：“帮帮我，我需要一些信息”
- 第二种：“帮帮我，我有麻烦了”
- 这两种需要都应该在设计帮助系统时考虑到
- 在帮助系统需要不同的功能



# 帮助信息

---

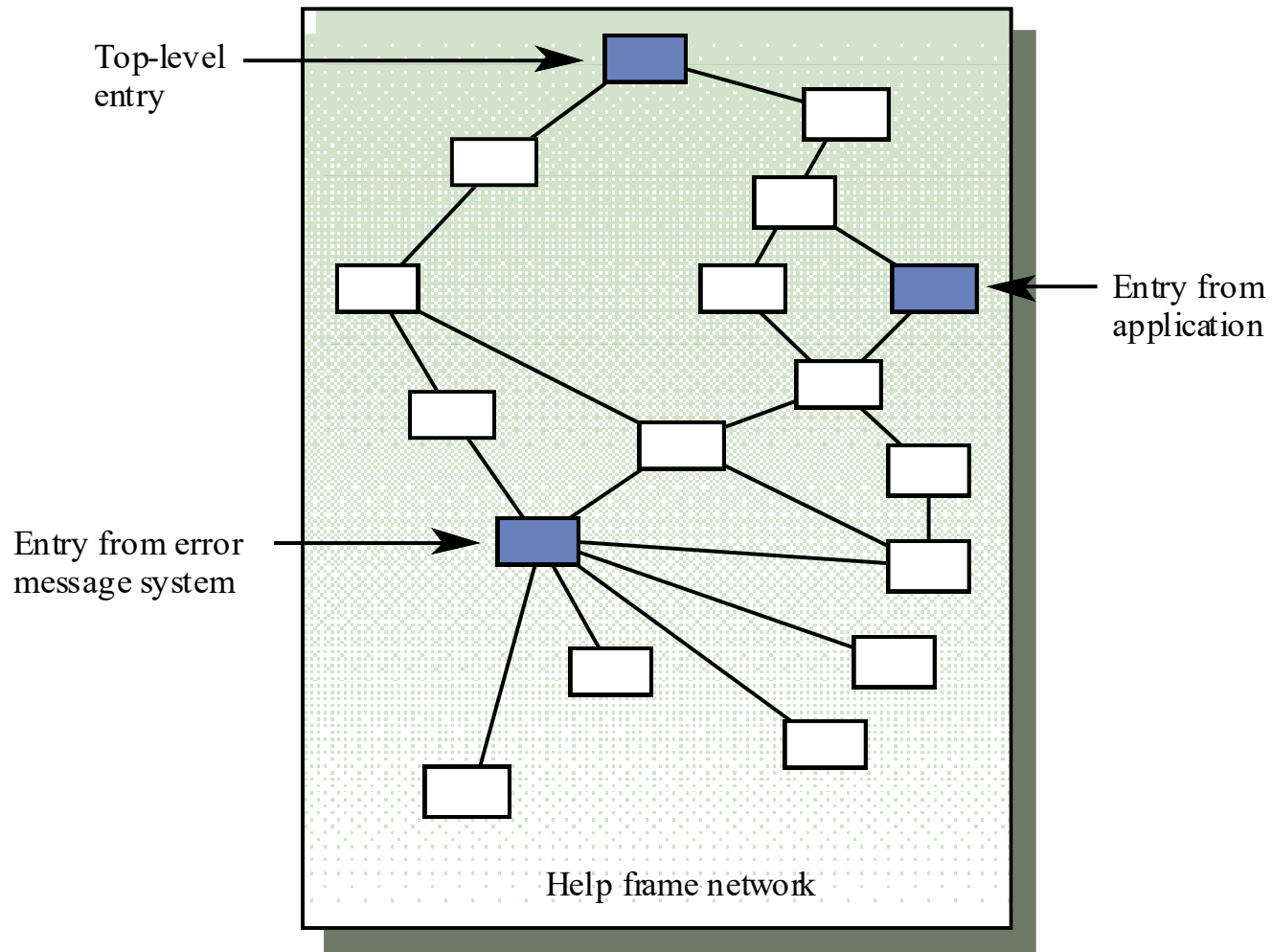
- 不能是简单的在线手册
- 屏幕和窗口跟纸张不能完全对应
- 显示器的动态特征可以提高信息表示的能力
- 人们在不习惯在屏幕上大量阅读文本

# 帮助系统的使用

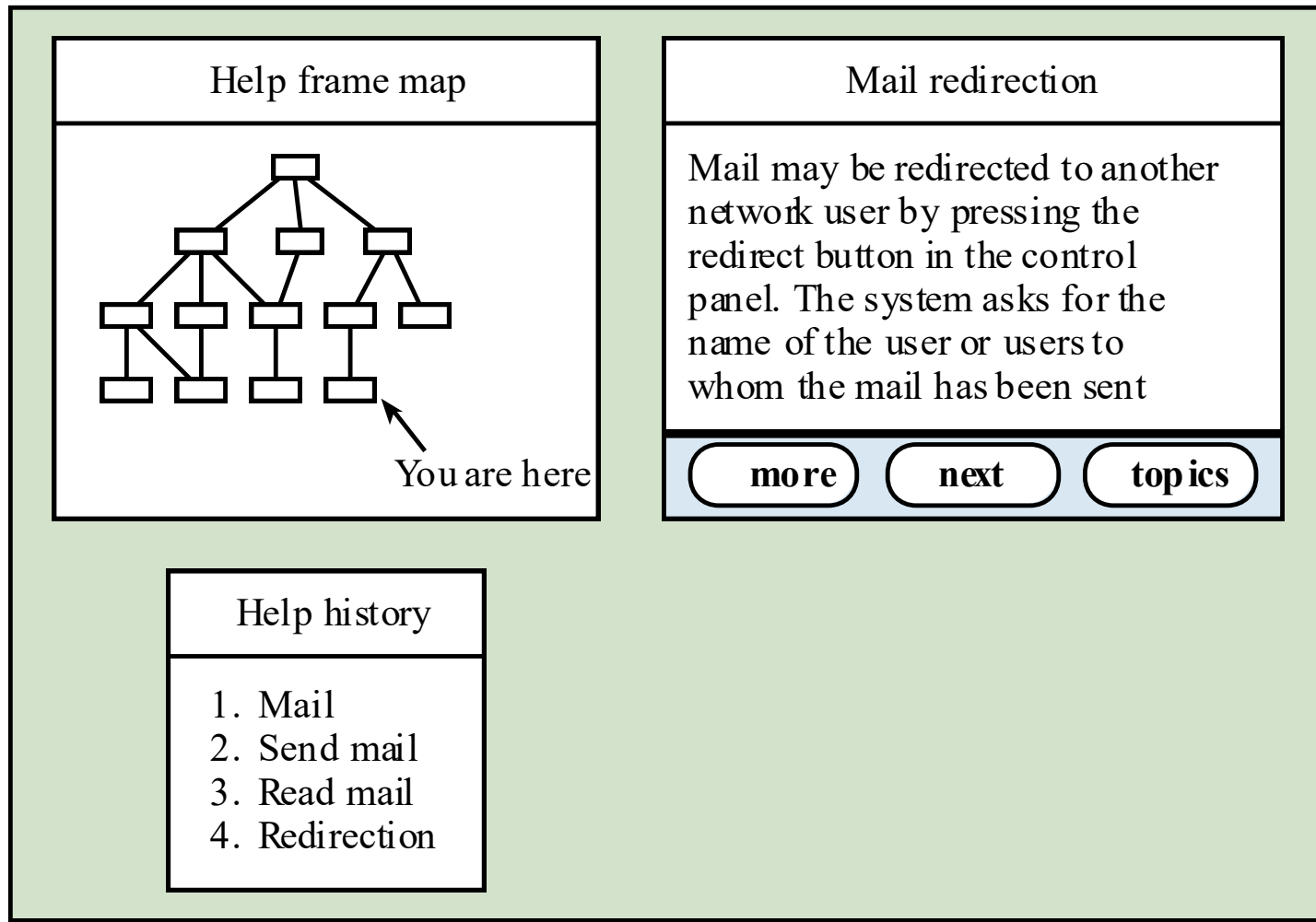
---

- 应该提供多个不同的入口，这样用户可以从不同的位置进入帮助系统。
- 应该提示用户当前处于帮助系统的什么位置
- 应该提供功能以使用户可以在帮助系统中导航或者来回移动

# 帮助系统的入口



# 帮助系统窗口



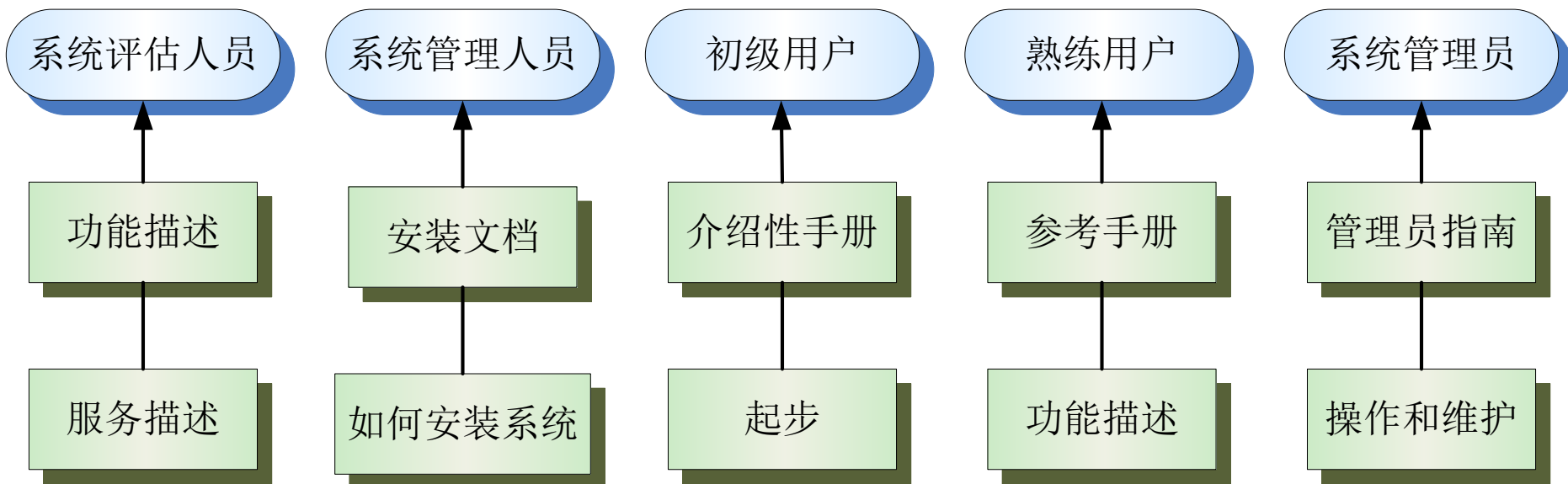
# 用户文档

---

- 提供在线信息的同时，纸质文档也应该提供
- 文档应该考虑到不同的用户，包括有经验的和没有经验的
- 在手册的同时，应该提供其他易于使用的文档（如快速参照卡片）

# 用户文档类型

---



# 文档类型

---

- 功能描述
  - 简要描述系统能做什么
- 介绍性手册
  - 系统的一般性介绍
- 系统参考手册
  - 详细描述系统的功能
- 系统安装手册
  - 描述如何安装系统
- 系统管理员手册
  - 描述如何管理系统

# 用户界面的评价

---

- 对用户界面进行评价，以评估其是否合适
- 对大多数系统来说，完全的评价是很昂贵的也是不切实际的。
- 理想的是，根据一份可用性描述来进行界面评价。然而，这样的可用性描述一般不具备。



# 可用性属性

---

属性	描述
可学习性	一个新用户需要多长时间才能成为一个系统熟练用户
操作速度	系统响应与用户工作情况的匹配程度如何
鲁棒性	系统对用户错误的容忍程度如何
可恢复性	系统从用户错误中恢复的能力如何
适应性	系统与单一工作模式结合的紧密程度如何

# 简单的评价技术

---

- 用户反馈的问卷调查
- 观察在系统使用过程中用户的表现
- 在软件中嵌入代码以收集功能使用和用户错误的相关信息
- 提供在线用户反馈的功能

# 要点

---

- 界面设计应该以用户为中心。界面应该具有逻辑性和一致性，应该帮助用户从错误中恢复。
- 交互方式包括：直接操作、菜单系统、表格填写、命令语言和自然语言
- 要表示变化趋势和近似值时，应采用图形化显示。要表示精确值时使用数字显示。
- 颜色使用应该保守一些，并保持一致

# 要点

---

- 系统应提供在线帮助，包括提供信息、解决困难。
- 错误消息应该是积极的而不是消极的。
- 应提供一系列不同类型的用户文档。
- 如果可能的话，应对照可用性属性的描述进行界面评价