GUI and HCI design

Goal is for software to improve productivity (correct plans & actions)

User Model
  cognition
  error
  individual differences (adaptive / learning models, AI ?)

Task Model   (not part of this presentation, HCI Software Engineering)
  definition & frequency
  strategies and operations

System Model (GUI + application)
  ease of use - learning
  customization
  power – support skilled performance
  robustness - reliable, error handling, help
GUI Design

User

Strengths

- World Knowledge
- Learner
- Pattern Matching
- Analogical
- Productive Thinking
- Vision & Sound

Software

Strengths

- Fast Accurate
- Reproductive "Thinking"
- Never Forgets
- Non Ambiguous Knowledge

Weakness

User

- Limited Awareness
- Accurate
- Reproductive Thinking
- Forgets

Software

- Limited World Knowledge (?)
- Not Analogical
- Poor Learning
- Poorer Input Senses (?)
User Model

Cognition = perception + memory

Software use is a cognitive / problem solving activity.

Users solve learned problems (skill) and new problems (analogy, generalization).

To solve problems users must perceive (recognize) them and understand them.

Human perception is pattern oriented.

We see the gestalt (and suffer illusions).

Human knowledge is procedural, episodic, and semantic.

- **procedural**: serial tasks
- **episodic**: individual life experiences
- **semantic**: knowledge, cultural
User's Syntactic Knowledge

Task and environment specific knowledge.

Syntactic knowledge facts are often discrete and disjoint from other syntactic facts.

**Learning**: arbitrary nature often requires rote learning, learn by doing.

**System dependency**: syntactic rules vary with system. Same goal requires different operations.

**Interference**: same operations can have different results across applications and systems.

**Reduce Syntactic Complexity**: structured command sets, menus, direct manipulation environments.
User's Semantic Knowledge

Conceptual knowledge about the domain of a task and environment..

Concepts are built upon each other they are interconnected and have some "semantic" structure -- relationship.

Semantic knowledge is best taught by analogy, or example, to other knowledge and by practical experience.

  Pictorial representations are helpful.
  Negative examples (misses).

Task experts maybe computer novices & computer experts maybe task novices.

Concepts: stable memory, generalizable across computer systems and applications.

Tasks: often decomposable into subtasks with analogy to other known tasks.
Error a planned mental or physical activity that failed its intended outcome where the failure is not attributable to chance events.

Intention a specification of desired action, a goal. Intentions generate plans (schemas, actions) to achieve goal.

Mistake an error in intention (deficient judgement or inference).

Lapse a failure in storage of the intention.

Slip an error in execution of intention.
GUI Strenths

GUIs are a mixture of direct manipulation and menu based interface styles.

Objects in task domain are visible: often icons

+ planning is a recognition (not recall) task
  modeless or visible mode (greyed, disabled menu items)

+ low syntactic & semantics memory
  icons have semantics by analogy

+ spatial / visual tasks learned faster, memory retained longer

+ minimizes errors (slips)
GUI Weaknesses

User directly manipulates task object. Actions and results are visible, incremental and reversible (undo last step).

- Repetitive tasks maybe hard to combine or parameterize, as in command line.

- Iconic interfaces may suffer description errors, visual interferences.
GUI Affordances

Norman (1988) discusses the affordances of everyday things.

**Affordance** - the perceived and actual properties of a thing.

When everyday things that have been used for decades or longer (doors, switches, knobs, ...) vary in their affordances - what is the affordance of software?

Commercial GUIs have a policy (look and feel) defining GUI affordances.

Commercial GUI development libraries and tools assume a GUI policy.

Design Question: What are the affordances of the menus, toolbars, icons, cursors and other UI controls in your interface? 
   
   examine the icons on your desktop or task bar (Mac's dock) 
   did you know what they represented the 1st time you saw them?
Affordances of UI controls & objects

UI Controls should indicate their state - are they affordable.

   Menu items inactive: invisible or grayed, active: checked.

Icons, iconic buttons, or graphic menu items suggest the functionality they afford.

Cursors indicate state. I beam enter text, hand to move, crosshair graphic editing, clock \( \parallel \) hourglass for activity in progress ...

Most likely next operation should be highlighted

On a dialog the most common response should be selected for user acceptance

Objects should indicate their selection.

   Selected Objects should be highlighted.
Command Buttons - Toolbars

Task-primary application interactions should occur in the application's window.

Getting, setting and saving properties and resources (files) should be done w/ menus and dialogs.

Menus

Limit menu choices (» 8) pulldown - popup.
  binary choices consider label rendering style ( B I U  in powepoint )

Order, or group menu choices when possible.
  use separators
  use redundant coding:  icon & text

Use walking (cascade) menus to show menu traversal path.

Provide mnemonic on all frequent menu items and accelerators for skilled performance on selected items.
Dialog Boxes

Dialogs break the user's pace -- they are special cases.

Dialog boxes are for complex and infrequent user interaction.

All dialogs should be movable (caption bar).
  Dialog caption should reflect purpose/action
  Dialogs should be as small as possible.

User errors report via dialogs (display usage and offer help).

Use the appropriate modality: Visual differentiate dialog types
  modeless doesn't halt use of the windows
    usage, help, howTo – user needs to interact w/ app & dialog
  application modal halts use of application
    Most common and appropriate modality -- warnings, input.
  system modal halts use of all applications
    use only when critical -- very rare use (System Utilities) !
Display User Wear (usage pattern) ??

Physical objects display wear from their use. The more use, the more wear...

Examples of wear affordances in reference manuals: "dog eared" pages are most used book opens to pages most often read book marks allow quick

standard scrollbar: thumb position

edit wear on text file width of histogram indicates magnitudes of edit on that line

Two edit histories (two histograms - group work)

Total edit wear of file

Edit wear of different edit operations.
## Table Lens

<table>
<thead>
<tr>
<th>Year</th>
<th>Product</th>
<th>Quarter</th>
<th>Channel</th>
<th>Units</th>
<th>Revenue</th>
<th>Profits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1993</td>
<td>ForeCode Pro</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1992</td>
<td>ForeWord Pro</td>
<td>539</td>
<td>VAR</td>
<td>1</td>
<td>226</td>
<td>79</td>
</tr>
<tr>
<td></td>
<td></td>
<td>540</td>
<td>Retail</td>
<td>16</td>
<td>3200</td>
<td>961</td>
</tr>
<tr>
<td></td>
<td></td>
<td>541</td>
<td>Retail</td>
<td>12</td>
<td>2400</td>
<td>720</td>
</tr>
<tr>
<td></td>
<td></td>
<td>542</td>
<td>Retail</td>
<td>5</td>
<td>1000</td>
<td>300</td>
</tr>
<tr>
<td></td>
<td>ForeMost Server</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ForeMost Lite</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ForeMost Access</td>
<td>756</td>
<td>VAR</td>
<td>761</td>
<td>684900</td>
<td>287658</td>
</tr>
<tr>
<td></td>
<td></td>
<td>757</td>
<td>VAR</td>
<td>475</td>
<td>427500</td>
<td>179550</td>
</tr>
<tr>
<td></td>
<td></td>
<td>758</td>
<td>VAR</td>
<td>428</td>
<td>385200</td>
<td>161784</td>
</tr>
</tbody>
</table>
Are recent “flat”, “minimal” GUI designs the counter argument to display wear?
Visual Lies

Visual lies usually overstate their data.
The lie factor $= \frac{\text{size of effect in graphic}}{\text{size of effect in data}}$

size of effects as percentages

18 m/g $= 0.6$ inches

27.5 mpg $= 5.3$ inches

53 % increase mpg
783 % increase line lengths

lie factor $= 14.8$
California Pension Retirement System's "pension buck"
California State Retiree, V XXVIII, No. 03, pg 6, 3/14/2014

<table>
<thead>
<tr>
<th></th>
<th>graphic (cm)</th>
<th>graphic label</th>
<th>lie factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>investment</td>
<td>7.9</td>
<td>65%</td>
<td>0%</td>
</tr>
<tr>
<td>state</td>
<td>2.6</td>
<td>21%</td>
<td>-3%</td>
</tr>
<tr>
<td>employee</td>
<td>1.7</td>
<td>14%</td>
<td>7%</td>
</tr>
</tbody>
</table>
3D chart, 1D data

Actual data: height of the bars = numerical value.

No information in 3D.

Attention drawn to the depth of the bars and the underlying grid.

Depth offsets vertical axis labels from height of bars.

Which is easier to estimate the value (height) of the bars?
Useful Junk?

Two recall tasks: after five minutes Vs. after two to three weeks.

Both cases, participants had a significantly higher chance of getting the message of the chart in the case of the highly embellished Holmes-style chart.

No significant differences in recalling the subject, categories, and trend between the two chart types after five minutes.

All three were significantly better for Holmes-style charts in the long-term case.

Bateman et al, SIGCHI, 2010
Mosaic charts – stacked area

Titanic
Women and children first
All children survived in 1st and 2nd class.

Mosaic data is percentages

<table>
<thead>
<tr>
<th></th>
<th>Class</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>Crew</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Survived</strong></td>
<td><strong>Gender</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>Adult</td>
<td>118</td>
<td>154</td>
<td>387</td>
<td>670</td>
</tr>
<tr>
<td>Yes</td>
<td>Adult</td>
<td>57</td>
<td>14</td>
<td>75</td>
<td>192</td>
</tr>
<tr>
<td>No</td>
<td>Child</td>
<td>0</td>
<td>0</td>
<td>35</td>
<td>0</td>
</tr>
<tr>
<td>Yes</td>
<td>Child</td>
<td>5</td>
<td>11</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td>No</td>
<td>Adult</td>
<td>4</td>
<td>13</td>
<td>89</td>
<td>3</td>
</tr>
<tr>
<td>Yes</td>
<td>Adult</td>
<td>140</td>
<td>80</td>
<td>76</td>
<td>20</td>
</tr>
<tr>
<td>No</td>
<td>Child</td>
<td>0</td>
<td>0</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>Yes</td>
<td>Child</td>
<td>1</td>
<td>13</td>
<td>14</td>
<td>0</td>
</tr>
</tbody>
</table>
Visual GUIs design goals

- Communication via Visual Language
- Functional > Aesthetic
- Simplicity > realism
- Geometry: scale, contrast & proportion
- Management: visual organization
- Grids: modular visual design
- Semiotics: icons & symbols as signs
- Style

Visual language:

- vocabulary: visual elements
- syntax: usage rules for elements
- literacy: experience of designer & user
- style: skill w/ vocabulary & syntax
Elegance and Simplicity

Reduce design elements to a minimum
• functionality not photo-realism
• reduce visual search (and cognitive) load

Simple designs are more: approachable recognizable, remembered usable (immediately and thereafter)

Reduction

Determine essential qualities(adjectives) -- color, labels, controls

Is each element needed? Would the design suffer if removed?

Test element's necessity by removing it. If design is fine omit element.

Omitted elements can be indirectly accessible via menus -- option buttons.
Leverage Design Rules /Elements

Find multiple elements doing similar functions

Design a combined element for the functions

Do not overload the modality of elements.

Design trade off between simple interface and cognitive load (rules for use).

---

display complexity

user's mental load

---

Cognitive Load

few rules

many elements

optimum

rules elements

many rules

few elements
Regulation
Use regular shapes, simple contours, muted colors
Make multiple similar forms visual properties identical (size, shape, alignment, spacing ...)
Limit font variation to few sizes in two families
Do not regularize critical elements -- make them stand out (novel)
Geometry: Scale, Contrast & Proportion

**Scale** is the element's relative size (area)

**Contrast** is an element's distinctive dimension. How is it differentiated from others?
{size, value (greyscale), hue (color), position, orientation, texture, shape}

**Proportion** is the ratio of sizes
- Computer displays 1.33 to 1
- Golden (classical) rectangles 1.618 to 1

Scale and contrast are used to emphasize elements.

**Activity** is how a design uses geometry to lead the visual search (view).
Humans seek patterns, the design should provide cues to group common, differentiate unique elements, and provide comparison (evaluation) information.
Visual Variables

Associative: independent of other variables. Most variables are associative.

size and value are dissociative - they affects visibility of other variables. (e.g. line to thin to see color)

Selective Perception: viewing isolates all group members into an image. shape is not selective.

Ordered Perception: viewing can determine ordering (ranking).

Ordering reduces need for legends (keys).

Position, size and value enable ordering

Quantitative Perception: viewing can determine relative amount of difference not just ranking. Must be easily apparent.

size (area) and position are quantitative
GUI Design

3 interacting variables of color vision:

- **Hue** color
- **Brightness** intensity (bright - dull)
- **Saturation** %color in field

Opponent process theory of color vision

These colors can't be seen in same patch of light (adjacent).
They produce shadows and edges.
Avoid use of opponent (opposite) colors.

Blue is the hardest color to see small changes in hue

Selection / Applicability

Color is very useful to have user selected items stand out in a display.
Color can also be used to indicate whether a menu option is valid in the current state or not ("greyed options").
Alert / Attention. Change of color represents change in state or mode (Traffic signals: green, yellow, red).

Use few colors that are easy to discriminate
Use warning colors sparingly.
Consistent system wide analysis of color use.

Element Discrimination. Color provides contrast and improves discrimination. Need high contrast difference. Contrast a function of luminance or hue.

Category grouping & field definition. Color can help group display elements and facilitate visual search. Visual search is affected by:

- number of items
- color separation of categories
- legibility of coded symbols
- relationship between color coding and targets
Size & Visual Acuity

As screen density increases color effect increase.

Color can define visual fields on display - weather maps

As number of colors increase size of text should also increase.

Color can't be assumed! Redundantly code display.

Designer's color perception != user's color perception

  color & text codes (categorization)
  color, size & text
  color, size, text & icon ....

Color Memory: 5 - 7 color memory for codes. Don't tax Working Memory use around 4 colors!
Shape coding consistency can be used to define an element's operation (function)

check vs radio buttons

Contrast coding can differentiate elements roles:
- pseudo 3D sunken field – data entry
- neutral field - label
- raised field - control (button)

Use **strong** contrast and **few** contrasts (one or few dimensions).

**Groupings:** use of regions, tabs, ...

Scale and contrasts (hue, texture, value) can define visual regions
- use selective **and** associative variables
  (group w/o affecting visibility)

**Squint test** - close 1 eye, squint w/ other at entire display.
what is "seen at a glance", what is processed first is visually dominant
Grouping guidelines

Group into a small number \((7 \pm 2)\)

Rank the importance of the groups

Show hierarchical relations with size
Show non-hierarchical groups with hue

Maximize differences between groups Minimize differences within groups.

Using Perceptual Distinction

Determine range of variation (min .. max) for sizes, color dimensions.
Use logarithmic > linear scaling for discrimination.
At least double each level!
Balanced use of figure / ground

Determine and equalize the visual weight of figure / ground.

Use internal padding to surround the figure and separate it from borders. Spend valuable screen real estate for internal spacing!

Position the figure w/in the ground. Centering is usually most appropriate.

White (Negative) Space is not wasted
White space is needed for figure / ground integration. White space helps spatial separation / organization.

Increase white space around critical elements
Alignment and Visual Relationships

Elements should be aligned with boundaries and margins.

Alter size and proportion when needed to support alignment.
- Extend elements beyond margin wrt sharpness of adjacent angles.
- Greater the acuteness of angle the greater the extension
- Proportional fonts use optical adjustment.

Items not aligned to anything on display should be proportional to display.

Find true point of alignment, dimension of extent, or unit of spacing needed.
GUI Design

Visual Structure

Structure guides a user's visual activity -- their path across a display.

Structure helps grouping: Gestalt: sum > parts

proximity

similarity

continuity

area

symmetry

closure
GUI Design

**Activity, Grouping**

Spatial logic (activity) should be congruent with user's actions

Spatial organization or grouping ➔ explicit structure (bounding boxes)

Use symmetry balance visual structure

Identify symmetry axes:
- vertical ➔ horizontal

Balance information on both sides of axes

Center symmetry axes in display context

Note use (and non use) of GroupBox, negative space, and top-down, left-right spatial usage sequence
Grids: modularity in visual structure

Regular visual structures are predictable, flexible, and efficient.

Grids benefit design and provide "scalability" to a GUI application. As screens and dialogs increase in numbers a grid layout simplifies design and increases use.

Canonical grid layout enables 6, 4, 3, and 2 division of elements on a display.

Any remaining visible grid lines should be half intensity in final display.
GUI Windows are rectangular. grids are the first display layout considered.

Determine vertical unit that allows any two controls to be adjacent

When spacing of multi-line controls consider labels as controls (in layout)

Horizontal unit 3 x as wide as vertical unit

Use 5-7 column divisions of horizontal units
AC3D: novice 3D modeler GUI

10 menu
4 22 buttons
1 labels
5 buttons
15 buttons
8 labels
13 textfields
8 buttons
1 v scrollbar
3 labels
6 buttons
1 label
7 buttons
1 label
1 textfield
15 buttons
1 h scrollbar
3 labels
2 checkboxes

(2 times)
3 menu
4 buttons
1 label
1 i/o pane

(2 times)
3 menu
4 buttons
1 label
1 i/o pane

163 gui components
Blender: expert 3D modeler GUI

- regulation, proportion, visual variables
- grid structure, figure / ground
- activity pattern, alignment

? gui components
Semiotics -- Images as signs

Iconic sign
representation resembles object

Symbolic sign
viewer associates representation and object

Abstractness helps iconic generality
Realistic icons represent instances
Stylized (minimalistic) icons represent a class
Icon Selection

Use iconic representation when communication goal is concrete & familiar.

Can use symbolic representation for repetitive concepts (learned).

Use text for abstract or complex (subtle) representations (processes).

Avoid mixing textual, iconic and symbolic signs w/in image set.

Icon Refinement

Determine the level of abstraction.

Try simplifying shapes into regular geometric forms.

Try using negative space to determine contour.
Coordinating Icons

Use a similar perspective & point of view

Use a similar style of representation -- don't combine icons with symbols.

Use consistent size, orientation, layout, color, and visual proportion (weight / area) to each image. Grids help internal structure.

Use the same elements when possible in your image set \{lines, rules, textures\} -- limit the visual vocabulary.
Mastering Style

Read style guides -- learn the conceptual model from the user's point of view.

Respect the visual language of the style.

Learn the usage and methods of user customization -- fonts, color and how they can possible degrade the style.

Working across (with) many Styles

Develop a translation table across the style set

Extend the widget set to fill out gaps in the translation table.

Use menu and control mechanisms of the style.

Focus on high level orienting features -- keep similar structure when possible.
Design as a reflection

**Design Model**

**User Model**

**ideal system**

designer

user

**actual system**

designer

user
Assume 3 competitive applications (A, B, and C) that exhibit these performance as a function of use functions.

System "A" is the fastest to learn and the fastest to asymptote at a moderate level of performance.
System "B" is slow to learn and slow to asymptote at a low level of performance.
System "C" is slow to learn and slow to asymptote at a high level of performance.

Are there user's, or tasks, appropriate for each system?
Studies supporting consistency examine ease of learning and ease of transference from one system to another.

Internal consistency reduces conceptual load

External consistency reduces training

Assumption is that easy to learn is easy to use, and to keep on using ...

However,

Humans are fast learners

Systems are acquired to use for increased productivity
not only for ease of use (learning)

Skilled performance is important
Case against Interface Consistency
J. Grudin, CACM, 1989

Consistency is a major watch word in guides for developing good user interfaces.

Consistency $\neq$ perceived consistency

3 Types of Consistency

1. Internal Consistency. Easily controlled by the design team. Command language

2. User Interface Consistency. Externally consistent with environment or desktop. -- User Interface controls menus, buttons, dialogs ...

3. Tool / Analogy Consistency. Consistent with a model, analogy, or external tool. -- Remote monitoring & physical gauges
Consistency is One of Many Goals

Intensive task use analysis of users is fundamental to user interface design.

Consistency can be centered on:

• what is most often done
• what did the user do last
• what would the user next next

Grudin's Arrow Keys

Which is better?

1 more consistent with external analogy
   - correction is opposite key
2 best performance
   + easy correction, multi-fingered use
**UI Testing**

Specify testing & validation at design time. Have internal and external testers.

What about the UI is confusing to the user?

Do users see, comprehend, use the following design elements:

- the use model, the design analogy
- the activity flow, visual structure
- the layout groupings
- the critical elements
- labels of UI elements
- meanings of images

With errors
  - does help actually help

With usage
  - do users improve productivity
References


Resources


Marathon Integrated Testing Environment, MarathonITE, is an affordable, easy-to-use and cross-platform Java/Swing™ GUI Test automation framework, http://marathontesting.com/, viewed 2/25/2014

