First Principles

GUI Bloopers Ch1

#1: Focus on Users and Task, not Tech

- Human-Centered Design (DOET)
- For whom is the software designed?
    - Users
    - Customers (not always the users)
- What activity does the software support?
- What are the skills and knowledge of your users?
- How do users conceptualize the data they interact with via software?
- Work preferences?

The Process of Understanding Users

- Business Decision
  - Who are the users?
  - What tasks to support?

- Empirical Investigation
  - Develop User Profiles/Persona
  - User Ability
    - General Computer Knowledge, Task Knowledge, System knowledge
  - Task Analysis

- Collaboration
  - Bring users into the development process/team

Understanding Tasks

- What user tasks are relevant to the application’s function?
- Commonality and importance
- Task steps and results/output
- Where do users get task-related information and how is it used?
- Who performs the tasks?
- What tools are used?
- Challenges to task completion
- Task terminology
- Relationships between tasks
- Communication required for tasks

Software Exists to Serve a Larger Purpose

- Context of use matters!

VS.

#2: Consider Function First, Presentation Later

- **DOES NOT MEAN** "implement the functionality first and slap a user interface on when it's done".

- **DOES MEAN** understanding:
  - What concepts will be visible to users?
  - What data will they create, view, manipulate?
  - What options, choices, settings, and controls will be provided?

The Conceptual Model

- An abstract explanation of a software’s function and what needs to be understood before using it.

- Focus on mapping functionality closely to user tasks.

- The cost of concepts:
  - Novel concepts are not readily recognized and must be learned.
  - Concept interactions in the software cause exponential growth in system complexity.

Building the Conceptual Model

- Objects/Actions Analysis
    - What conceptual objects to expose to the user
    - Permitted actions on the object
    - Attributes (settings of the object)
    - Relationships between objects
        - Type hierarchy: an *“is a”* relationship
        - Part/whole hierarchy: a *”part of”* or *“contains”* relationship
- A lexicon of consistent terminology used in software and documentation
- Task scenarios

#3: Conform to the users’ view of the task

- Striving for “Naturalness”
  - This aligns with domain expectations and the conceptual model
- Avoid things that are hard to learn and easy to forget
  - Extra steps that seem unnecessary
  - Arbitrary restrictions
- Use the lexicon to enforce your users' vocabulary
- Encapsulate program internals
  - How the software works isn’t important

Complexity Reduction

- Sensible defaults
- Template or “canned” solutions
- Guided path-wizards
- Progressive disclosure
  - Advanced features presented when needed
- Generic Commands
- Task-specific Design
- Customizability
  - Risky option

#4 Design for the Common Case

- Common goals should be the easiest to achieve
- What is “common”?

	By Most Users	By Few Users
Frequently Used	Highly visible; few clicks	Barely visible; few clicks
Rarely Used	Barely visible; more clicks OK	Hidden; more clicks

- Don’t worry about edge cases.
  - Takes development time and resources away from the common cases
  - Forces the UI to support all possible cases

#5: Don’t Distract Users from Their Goals

- Don’t add extra problems for the user to solve
- Software should support problem solving in the target task domain
- Minimize or remove the need to problem solve in the domain of computer technology
- Users should not have to reason by elimination
  - Features should be obvious

#6: Facilitate Learning

- Focus on outside-in thinking
  - Users don’t know what you mean
  - Knowledge in the world can guide them
- Ambiguity is the enemy
- Focus on the conceptual model
- Consistency
  - Helps to build habits
  - Difficult to implement
  - User-centered view allows for predictability
- Tolerance for Mistakes/Errors (low-risk)

#7: Deliver Information not Just Data

- Data becomes information when processed, organized, and interpreted to provide meaning and context
- The presentation of information is critical
  - Visual Order and Focus
  - Scannability
  - Match the medium
  - Attention to detail
- The screen belongs to the user
  - Do not ”take” controls, move, or rearrange data without user permission
  - Changes to the display should be localized as much as possible

#8: Design for Responsiveness

- This does not mean a ”responsive” design for different screen sizes.
- Focuses on perceived speed
- A requested action may take time
  - Acknowledge an action immediately
  - Provide feedback regarding the status
  - Indicate a “busy” status
  - Allow the action to be cancelled
  - DO NOT BLOCK THE UI!

#9: Try it out on users, then fix it

- We all are worried to show people things that aren’t “done”, ”perfect”, or ”in progress”.
- Software is never really “done”!
- Let people see and try it.
- Take feedback and use it to improve the software.
- Make user testing a “safe” event and activity.
  - Be a passive observer.
  - It isn’t personal! ☺
  - Without feedback, how do we know we’re making the “right” product?