The design of everyday things
(Norman, 1990)

The ordinary objects reflect the problems of user interface design
- Door handles
- Washing machines
- Telephones
- etc.

Introduces the notion of affordance, metaphores, and conceptual models

Provides design rules
Conceptual model vs. mental model

- Conceptual model: formal, structured, logical
- Mental model: informal, incomplete, sometimes erroneous

Image of the system

Designer

User
Metaphor

Transfer of a relationship between a set of objects to another set of objects in a different domain

office/desktop

folders

electronic desktop
Affordances
Affordances

Quality of an object, which allows a user to perform an action

The form, the size, the view of the object suggest what we can do with it

« Much of everyday knowledge resides in the world, not in the head » (Norman, 1988)
Affordances

Dials for turning

Sliders for sliding
Affordances

Button for pressing but action unknown

These buttons?
Affordances

The concept of affordance was first introduced by psychologist James J. Gibson in 1977.

Gibson’s affordances are independent of the individual’s ability to recognize them. They depend on their physical capabilities.

Norman’s affordances also depend on the individual’s perception. Norman explained that he would rather replace his term by the term « perceived affordances ».
Perceived Affordances in this UI?
Our mental models of the mechanics and physics help us predict and simulate the operation of an object.
Constraints

Are these user interfaces effective?
Mappings

Example: Find the correspondance between the stove burners and the controls
Mappings

Example: Find the correspondance between the stove burners and the controls

...and now?
Example: designing a watch

Conceptual model?

Affordances?

Mappings?
Example: designing a watch

Conceptual model?

Affordances?

Mappings?

...and user feedback?
Norman’s principles (1990)

1. Make things visible
   We can know the state of a system by observing the user interface

2. Principle of mapping

3. Principle of feedback
   Inform the users about the state & result of their actions
Usability

« The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use »

(ISO 9241)

A usable system is: easy to learn, easy to memorize, efficient, visually appealing and fast to recover from errors
Utility vs. Usability

Is D better than A? What do you think?
Usability principles (Nielsen 2001)

- Visibility of system status
- Match between system and the real world
- User control and freedom
- Consistency and standards
- Help users recognize, diagnose and recover from errors
- Error prevention
- Recognition rather than recall
- Flexibility and efficiency of use
- Aesthetic and minimalist design
- Help and documentation
Visibility & feedback

Objective: aid the use and learning of a system

*Feedback* and *feed-forward* mechanisms to

- reduce memory load
- prevent errors (more later)
- reassure (e.g., progression of an operation)

helps user understand
  - what actions are available
  - what the system is doing
  - how it is interpreting the user’s input

... users should always be aware of what is going on
Recommendations: feed-forward
   gray out non-available commands
   make input possibilities clear
   give list of possible inputs instead of typing
   give example of expected input
   give intelligent default values

Recommendations: feed-back
   each user action should be followed by a changed representation in the interface
   inform users of long operations
   indicate currently used modes
   show status of system operations in progress
Visibility & feedback

System Response time (time to give feedback)

how users perceive delays

< 0.1s    perceived as “instantaneous”

1s    user’s flow of thought stays uninterrupted, but delay noticed

10s    limit for keeping user’s attention focused on the dialog

> 10s    user will want to perform other tasks while waiting
Visibility & feedback

Dealing with long delays

Cursors for short transactions

Percent done dialogs
  time/work left
  estimated time

Random for unknown times
Visibility & feedback

Currently used modes

What did I select?

What mode am I in now?

How is the system interpreting my actions?
Match between system and real world

The system should be integrated in user activities

Recommendations :

- speak the user’s language
  - e.g., informative messages
- information coherent with respect to other tools the user uses
  - e.g., electronic version of a paper form
- access to commands compatible to user’s task
  - e.g., frequent commands more visible, order of windows

Need to study and analyze user work practices
Match between system and real world

Use meaningful mnemonics, icons & abbreviations
  e.g. File / Save
    Ctrl + S  (abbreviation)
    Alt FS    (mnemonic for menu action)
               (tooltip icon)
Match between system and real world

Be as specific as possible about operations, based on user’s input

Best within the context of the action
Match between system and real world

Good use of metaphors and transfers

From Microsoft applications
User control and freedom

Users don’t like to feel trapped by the computer! They should offer an easy way out of as often as possible.

Strategies:
- Cancel button (for dialogs waiting for user input)
- Universal Undo and Redo (can get back to previous state)
- Interrupt (especially for lengthy operations)
- Quit (for leaving the program at any time)
- Defaults (for restoring a partially filled form)
- ...consider autosaving
Consistency & standards

Global coherence of interface
   internal: inside the application
   external: between applications (e.g., icons, shortcuts),
           w.r.t. the metaphor of the system (e.g., desktop)

Principle: a system that seems familiar is seen as easy to use by users

Goal: help learning and use

Risk: block system evolution (rigidity of standards)
Consistency & standards

Recommendations
  windows should look similar
    e.g., search box at top right
  consistent graphics
    e.g., information/controls in same location on all windows
  same vocabulary used for commands as other systems
    e.g., open / copy-paste / preferences / ...
  syntax of commands coherent across all the interface
    e.g., similar actions have similar effects

Consistency is not only visual consistency
Other examples: syntax, interaction, command result
Consistency & standards

Style guides:
published by system designers
describe the look and feel of a platform
are often too strict: help those who follow them and make life difficult for anyone who wants to deviate …

Examples:
- Apple Human Interface Guidelines
- iOS Human Interface Guidelines
- MS Windows Design Guidelines
- Android Design Principles

In principle good, but can be hard to follow
Implemented (in part) in interface toolkits
Error prevention

Even better than good error messages is a careful design which prevents a problem from occurring in the first place.

Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
Error prevention

Prevent errors: try to make errors impossible

Provide reasonable checks on input data
e.g., if entering order for office supplies
500000 pencils is an unusually large order. Do you really want to order that many?
**Error prevention**

**Mode errors**

- do actions in a mode thinking you are in another
  - refer to file that's in a different directory
  - look for commands / menu options that are not relevant

minimize by

- have as few modes as possible (or none)
- make modes highly visible
Error recovery

Help users recognize, diagnose, and recover from errors

Error messages should be expressed in plain language (no codes)

Precisely indicate the problem, and constructively suggest a solution.
Error recovery

Prevent/mitigate continuation of wrongful action:

**Gag**

deals with errors by preventing the user from continuing
e.g., cannot get past login screen until correct password entered

**Warn**

warn people that an unusual situation is occurring
... when overused, becomes an irritant
e.g., audible bell, alert box
Error recovery

**Do nothing**
illegal action just doesn’t do anything
user must infer what happened
  - e.g., enter letter in numeric-only field (key clicks ignored)
  - e.g., put a file icon on top of another file icon (returns it to original position)

**Self-correct**
system guesses legal action and does it instead
but leads to a problem of trust
  - e.g., spelling corrector
Error recovery

**Lets talk about it**

- system initiates dialog with user to come up with solution to the problem
  - e.g., compile error brings up line in source code

**Teach me**

- system asks user what the action was supposed to have meant
- action then becomes a legal one
  - e.g., adding a word in the spelling dictionary
Error recovery

If all else fails provide meaningful error messages
error messages should be in the user’s task language
don’t make people feel stupid

Try again, bonehead!
Error 25
Cannot open this document
Cannot open “chapter 5” because the application “Microsoft Word” is not on your system
Cannot open “chapter 5” because the application “Microsoft Word” is not on your system. Open it with “OpenOffice” instead?
Problematic error messages

Adobe's *ImageReady*

AutoCAD *Mechanical*

*Windows Notepad*

Microsoft's *NT Operating System*
Recognition rathen than recall

Computers good at remembering, people not!
Promote recognition over recall
menus, icons vs text commands, field formats
promote visibility of objects (but less is more!)

From Microsoft applications
Recognition rathen than recall

Give input format, example and default
Reducing memory load

Small number of rules applied universally

Generic commands

- Same command can be applied to many objects
- Interpreted in context of interface object: copy, cut, paste, drag ‘n’ drop, etc. for characters, words, paragraphs, circles, files

Contextual menus
Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users.

Allow users to tailor frequent actions.
Flexibility & efficiency of use

Capability to adapt to different contexts of use

Recommendations:
- permit command activations from keyboard or mouse
- allow frequently used operations to be activated by every location
- allow users to parameterize their software based on their preferences
- give quick access to frequent commands in menus

Can contradict minimalist design (later)
Expert users - want to perform frequent operations quickly

**Strategies:**
- keyboard and mouse accelerators/shortcuts
  - abbreviations
  - command completion
  - context menus
  - function keys
  - double clicking vs menu selection
  - type-ahead (entering input before the system is ready for it)

- navigation jumps and search
  - e.g., going to window/location directly, avoiding intermediate nodes

- history systems
  - WWW: ~60% of pages are revisits
Flexibility & efficiency of use

- Keyboard shortcuts for menus
- Customizable toolbars and palettes for frequent actions
- Split menu, with recently used fonts on top
- Double-click raises object-specific menu
- Double-click raises toolbar dialog box
- Scrolling controls for page-sized increments

7. Provide shortcuts

Experienced users should be able to perform frequently used operations quickly

Strategies:
- keyboard and mouse accelerators
  - abbreviations
  - command completion
  - menu shortcuts
  - function keys
  - double-clicking vs menu selection
- type-ahead (entering input before the system is ready for it)
- navigation jumps
  - e.g., going to window/location directly, and avoiding intermediate nodes
- history systems
  - WWW: ~60% of pages are revisits
Aesthetic and minimalist design

Dialogues (windows) should not contain information which is irrelevant or rarely needed.

Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
Ways to reduce visual clutter and focus user attention

Recommendations (be concise):
- only display important information (for what the user needs)
- reduce number of actions needed to perform an objective
- minimize input and reading instructions
- avoid too much text
- don’t ask for input that you can infer automatically
- avoid users having to remember information
- don’t ask users to perform calculations
Aesthetic and minimalist design
Provide help and documentation

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation.

Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.
Provide help and documentation

Help is not a replacement for bad design!

Simple systems:
  walk up and use; minimal instructions

Most other systems:
  feature rich
  simple things should be simple
  learning path for advanced features
Many users do not read manuals prefer to spend their time pursuing their task

Usually used when users are in some kind of panic
  online documentation better
  good search/lookup tools
  online help specific to current context

Sometimes used for quick reference
  syntax of actions, possibilities...
  list of shortcuts ...
Provide help and documentation

Tutorial and/or getting started manuals

short guides that people are likely to read when first obtaining their systems
  encourages exploration & getting to know the system
  tries to get across essential conceptual material

on-line “tours”, exercises, and demos
  demonstrates basic principles through working examples
Provide help and documentation

Reference manuals
used mostly for detailed lookup by experts
rarely introduces concepts
thematically arranged

on-line hypertext
search / find
table of contents
index
cross-index
Provide help and documentation

Reminders

- short reference cards
  - expert user who just wants to check facts
  - novice who wants overview of system’s capabilities

- keyboard templates & icons
  - shortcuts/syntactic meanings of keys
  - recognition vs. recall

- tooltips and other context-sensitive help
  - text over graphical items indicates meaning or purpose
Provide help and documentation

Wizards
walks user through typical tasks
... but dangerous if user gets stuck

What's my computer's name?
Fred?
Intel?
AST?
Provide help and documentation

Tips

migration path to learning system features
context-specific tips on being more efficient
must be “smart”, otherwise boring and tedious

Also consider: You can drag a submenu that has a move handle (a bar at the top of the menu) anywhere on the screen to create a floating toolbar.
Provide help and documentation

Contextual Video Clips

Mac OS configuration for the trackpad
Evaluating the user interface
Why bother about evaluation?

Pre-design
- investing in new expensive systems requires proof of viability

Initial design stages
- develop and evaluate initial design ideas with the user
Why bother about evaluation?

Iterative design
- does system behavior match the user’s task requirements?
- are there specific problems with the design?
- what solutions work?

Acceptance testing
- verify that system meets expected user performance criteria
Naturalistic approach

Observation occurs in a realistic setting

Problems
- hard to arrange and perform
- time consuming
- may not generalize
Experimental approach

The experimenter controls all environmental factors

- study relations by manipulating *independent* variables
- observe effect on one or more *dependent* variables
- Nothing else changes

**Example:** Testing whether there is a difference in user performance (time & error rate) between typing or writing text with a pen.
Experimental results

Example of results for the movement time required to point to targets on the screen by using two different devices (Device A and B).

Here, the experimenter controls the difficulty of the tasks (computed as a function of the distance and size of the targets).

\[ MT = 548 + 420 \, ID \]

\[ R^2 = .9719 \]

\[ MT = 173 + 197 \, ID \]

\[ R^2 = .9849 \]
Trade-offs

Natural vs. Experimental

- precision and direct control over experimental design vs.
- desire for studying the use of the system in real life situations
Evaluation techniques

Informal and quick:

- Heuristics
- Heuristic Evaluation
- Design Walkthrough
- Others ...

Formal and targeted:

- Alternatives User Studies
- Controlled Experiments
- Quasi-experiments
- Others (Interviews, Questionnaires, Observations)
Design (cognitive) walkthrough

Goal:
Aid to informally and quickly identify problems, using evaluation criteria (to be defined by you in advance)

Procedure
Choose a small group with different expertise and roles
Fix the duration to 1h max
A presenter describes a scenario (storyboard, video prototype, system)
Choose levels of critiques
The group identifies as many problems as possible
Use rules to aid in problem finding
(e.g., design principles, specifications, usability criteria, task sequence)
Design walkthrough

Specific
  e.g., “it needs 3 steps to do a simple search”

Missing Functions
  e.g., “no help provided, need search widget”

Bugs
  e.g., “the import functionality does not work”

Suggestions
  e.g., “provide an overview of the data generated”

General (the least useful)
  e.g., “difficult to use, too many icons”
Usability principles (Nielsen 2001)
- Again

Visibility of system status
Match between system and the real world
User control and freedom
Consistency and standards
Help users recognize, diagnose and recover from errors
Error prevention
Recognition rather than recall
Flexibility and efficiency of use
Aesthetic and minimalist design
Help and documentation
Heuristic evaluation

Systematic inspection to see if an interface complies to a set of usability principles

Method

- 3-5 inspectors
- usability engineers, end-users, double experts...
- inspect interface in isolation (~1–2 hours for simple interfaces)
- compare notes afterwards
  - single evaluator only catches ~35% of usability problems
  - 5 evaluators catch 75%

Works for paper prototypes, interactive prototypes, working systems
Forms of inspection

Self-guided
- open-ended exploration
- Not necessarily task-directed
- good for exploring diverse aspects of the interface, and to follow potential pitfalls

Scenarios-based
- step through the interface using representative end-user tasks
- ensures problems identified in relevant portions of the interface
- ensures that specific features of interest are evaluated
- but limits the scope of the evaluation - problems can be missed
Is heuristic evaluation effective?

3-5 evaluators find 66-75% of usability problems
- different people find different usability problems
- only modest overlap between the sets of problems found
Usability study (or alternatives)

Observe people with systems in simulated settings
- people brought into an artificial setting that simulates aspects of real world settings
- people given specific tasks to carry out
- compare alternative designs
- observations / measures made as people do their tasks
- look for problems / areas of success
- good for uncovering ‘big effects’
Observing many users is expensive

...but individual differences matter
  - best user 10x faster than slowest
  - best 25% of users ~2x faster than slowest 25%

Partial solution
  - reasonable number of users tested
  - reasonable range of users
  - big problems usually detected with a handful of users
  - small problems / fine measures need many users
Ethics

Testing can be a distressing experience
- pressure to perform, errors inevitable
- feelings of inadequacy
- competition with other subjects

Golden rules
- subjects should always be treated with respect
- always explain you are testing the system, not the user
- explain how comments and criticisms are good
Ethics

Don’t waste the user’s time
- use pilot tests to debug experiments, questionnaires, etc.
- have everything ready before the user shows up

Make users feel comfortable
- emphasize that it is the system that is being tested, not the user
- acknowledge that the software may have problems
- let users know they can stop at any time

Maintain privacy
- tell user that individual test results will be completely confidential

Inform the user
- explain any monitoring that is being used
- answer all user’s questions (but avoid bias)

Only use volunteers
- user must sign an informed consent form