User Interface Overview

James Baldo Jr.

SWE 432

Design and Implementation of Software for the Web

What is Usability Engineering?

- Requires knowledge of some psychology – theory
- Uses graphics – not how, but what to do with it
- Depends on GUI programming

Usability engineering is about designing interfaces for the user

Usability Engineering

- A design class
- Engineers tend to focus on functionality
  - But slick features are worthless if users cannot use them
- VCR programming
  - Programming was impossible with the original interfaces
  - It’s easy with new ones

User Friendly

- The term user friendly is over-used and under-defined
  - What is "friendly" to one person may be trite, tedious, or confusing to another
- "User appropriate" is a much more meaningful term
  - But we have to know the user

Never use the term "user friendly" again!

- This class is largely about communication
  - Communication between software and people

Software Design

- Inside-out
  1. Develop a system
  2. Then add the interface
- Outside-in
  1. Develop the interface
  2. Then build the system to support it

When design decisions are made, either the developer must conform to the user, or the user must conform to the developer.

Software Design (2)

- Effective software systems could be designed inside-out in the 1970s
- Modern systems must be designed outside-in to be effective
- Web sites sink or swim based on the usability

Traditional computer science courses are almost entirely inside-out!

Fundamental Software Design Principle: the 7 ± 2 Rule

- Human's short term memory can only hold about seven things at a time (plus or minus 2)
- That is all we can concentrate on!
  - Sports
  - Books
  - People and organizations
  - Software
  - User interfaces
- When we get more than about 7 items, we get confused

Brain Washing

- When we use the same interface repeatedly, we get “blinded” to the usability problems
  - Familiarity breeds content

- We sometimes “brainwash” ourselves into not noticing the problems

- If you look at an interface and keep the fundamental principles of user interfaces in mind, then the brain washing doesn’t matter anymore
  - You do not see the interface as a whole, but individual pieces

Simplicity

- An old quote:

  "It's easy to make things hard, it's hard to make things easy"
- Or as Mark Twain said:

  "It takes three weeks to prepare a good ad lib speech"
- Simple is hard!
  - A good interface is a lot like a good umpire … you never notice it's there

Shneiderman's Measurable Criteria

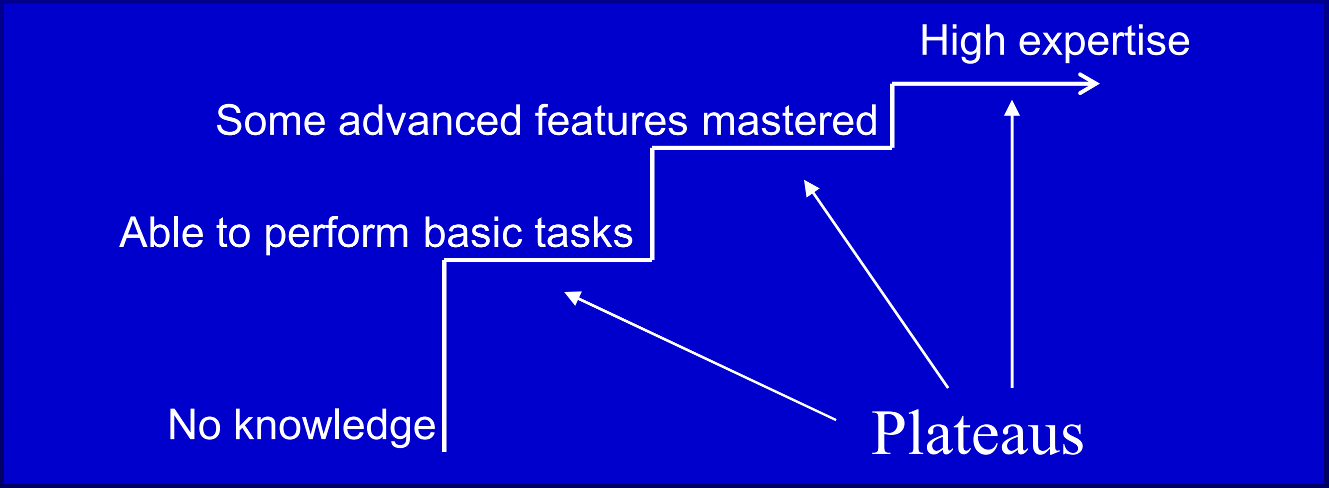
- User interface design has long been considered an art rather than a science
  - That is, decisions have been made subjectively rather than objectively
- There has been a lot of effort to make UI design more objective – that is, an engineering activity
- This course will teach you some of that
- The most important step was taken by Shneiderman …

Shneiderman’s Measurable Criteria (2)

1. Time to learn : The time it takes to learn some basic level of skills
2. Speed of UI performance : Number of UI “interactions” it takes to accomplish tasks
3. Rate of user errors : How often users make mistakes
4. Retention of skills : How well users remember how to use the UI after not using for a time
5. Subjective satisfaction : The lack of annoying features

1. Time to Learn

- With complicated UIs, the users must “plateau”



- Well designed interfaces make
  - the first plateau easy to get to
  - subsequent plateaus clearly available

2. Speed of UI Performance

- This is about navigating through the interface, **not** how fast the software or network runs
- *Interaction points* are places where the users interact with the software:
  - Buttons
  - Text boxes
  - Commands
- Speed of UI performance is roughly how many interactions are needed to accomplish a task

2. Speed of UI Performance:
The tyranny of the mouse

- The simplest way to slow down a UI is to use the mouse
- The mouse is incredibly slow: Most users can type between 8 to 15 keystrokes in the time it takes to move the hand from the keyboard to the mouse
  - The two activities use different muscles and parts of the brain
- Good UI designers need to reduce the amount of keyboard-to-mouse movements

3. Rate of User Errors

- Users will always make mistakes
- UIs can encourage or discourage mistakes
- Consider:
  - C/C++ : The lack of typing, particularly on pointers, and the complexity of the syntax actively encourages programmers to make mistakes. (Thus, we become debuggers, not programmers.)
  - Unix : The large, complicated command language encourages many mistakes as a result of simple typos and confusion.

4. Retention of Skills

- “Once you learn to ride a bicycle, you never forget”
- Some interfaces are easy to remember, some are hard
- If an interface is very easy to learn, then the retention is not important – users can just learn again
- Retention is typically more important with UIs that are hard to learn

5. Subjective Satisfaction

- Subjective satisfaction is defined to be how much the users “like” the UI
- This depends on the user (thus the word "subjective")
- Think of it in reverse: Users are dissatisfied when there is something annoying in the interface
  - Blinking
  - Ugly colors
  - Spelling errors in massages
- Most important in very competitive software systems

Tradeoffs Among Criteria

- There are always tradeoffs among the criteria
- Most people today equate "user friendly" with "time to learn" – this is a very narrow view of the world
- Making a UI easier to learn often winds up reducing the speed
  - Example: Many GUIs are easy to learn, but slow
  - Many command languages are fast, but hard to learn
- To be an effective UI designer, we must consider each criterion carefully and prioritize before designing

Establishing Criteria Priorities

Before designing, decide what is <u>acceptable</u> for each of the five criteria

- Order of priorities
- Minimally acceptable
- Optimistic goal

Three Categories of Knowledge

1. Syntactic Knowledge
   - Varied, dependent on computer and OS
   - Based on rote memorization
   - Easy to forget
2. Task Semantic Knowledge
   - Structured
   - Independent of computer and OS
   - More stable in memory

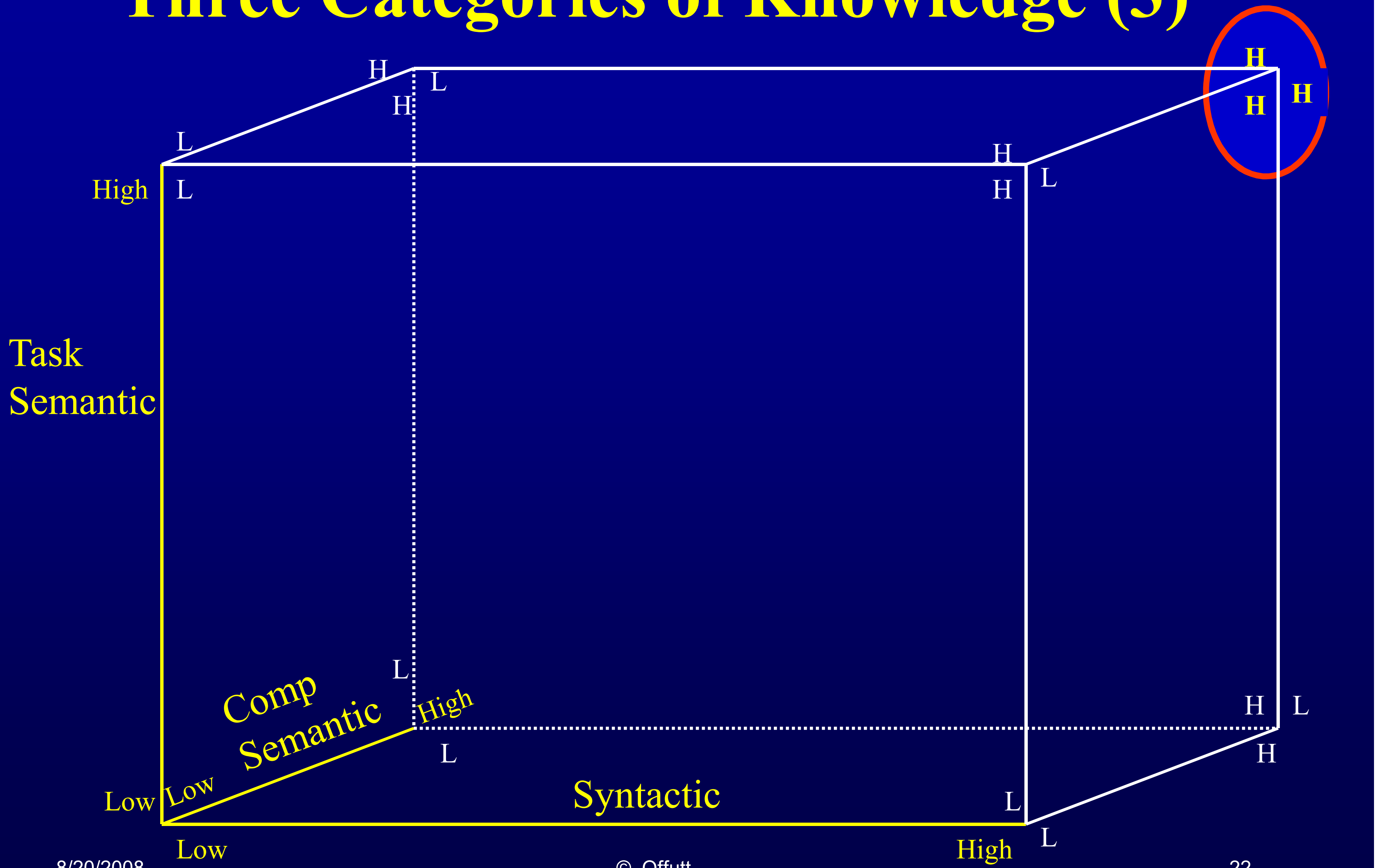
Applies here as well

3. Computer Semantic Knowledge
   - About how software and hardware works on the inside
   - Not learned by using software, but from reading and classes

Three Categories of Knowledge (2)

- Good syntax can:
  - Decrease the amount of memorization
  - Decrease time to learn
  - Decrease rate of errors
- Semantic knowledge involves:
  - Actions – things that can happen
  - Objects – things that exist

Three Categories of Knowledge (3)


H
H
H
H
L
H
L
High
L
H
H
L
Task
Semantic
L
Comp
Semantic
High
H
L
L
H
Low
Low
Syntactic
L
Low
High
L