Arguments in Favor of Usability
1. Major catastrophes
2. Minor catastrophes
3. The productivity paradox

The Productivity Paradox
- As the investment in information technology has gone up, the return on assets has actually been going down.
- The investment of 254 firms versus their return on assets. Firms that spent more of their income on computers had a slightly lower return on their

The Productivity Paradox
- In phase one, computers are used to automate mathematical, scientific, and routine tasks.
- Examples: Radar-directed gunfire, CAT scan X-ray and MRI, telephone switching networks, production-line robots.
- In phase two, computers work as digital "power tools" to augment human endeavors that cannot be completely turned over a machine.
- Examples: any task that require human activities such as speaking, understanding language, reading, writing, creating art, persuading, negotiating, organizing, socializing
- "Phase two helpers are not helpful enough to be worth their wages." - Landauer.

Building Useful Computer Systems
- Question: How can engineers design and build computer systems that are truly useful to people and will help people do real tasks?
- Answer: The same way they build other useful human-machine systems, by applying known methodologies and techniques from the field of human factors.

Human Factors
- Human factors discovers and applies information about human behavior, abilities, limitations, and other characteristics to the design of tools, machines, systems, tasks, jobs, and environments for productive, safe, comfortable, and effective human use. (Sanders and McCormick, 1987)

The Design of Everyday Things
- The vocabulary draws from psychology and human factors.
- The perspective come largely from the field of human factors, and provides a human factors perspective for the non-practitioner.
- Makes a compelling argument.

How can anyone get anything done?
- People use the information available in the appearance of objects, which Norman calls the Psychology of Everyday Things (POET).
- The designer makes operations clear, taking advantage of what people already know.
- Hence, designers must combine a knowledge of how-things-work with a knowledge of how people work and what they expect.

Basic Concepts of Design
- A good conceptual model allows us to predict the effects of our actions, and is necessary when things go wrong.
- The correct parts must be visible, and they must convey the correct message. Visibility is a major component in the mapping between the intended actions and the actual operations.

Norman's fundamental principles for designing for people:
- 1. Provide a good conceptual model.
- One more is needed: Test with real people doing real tasks.

+ Affordances.
  - What is the mapping between the perceived and actual properties?
  - Term is from Gibson; the psychological phenomena is "stimulus-response compatibility".
  - But many affordances are actually learned. Example: the holes on scissors invite fingers. What about the hole in a cigar cutter? Which way should faucet handles go? Which way do you screw in a lightbulb?
  - Cordless phone example: How do you turn the ringer on and off? Right now, is it on or is it off?
  - ATI example.

+ Conceptual Models
  - The designer has a conceptual model that guides the design of the device.
  - The device presents the user with a system image.
  - The user forms a mental model by looking at and interacting with the device.

- The seven stages of action
People Use Systems
To Get Work Done
- Main Ideas:
  - Most devices are harder to use than they need to be.
  - People often blame themselves.
  - People have knowledge, learned skills, expectations.
  - People have fundamental capabilities and limitations.
  - Good design takes these human characteristics into consideration.
  - Norman (1990) provides a perspective and a set of concepts for designing and evaluating devices based on these human characteristics.

Knowledge in the Head and in the World
- Precise behavior from imprecise knowledge
  - The world (devices and/or the environment around you) reminds you how to do things, and thus helps to direct the interaction.
- Examples:
  - Giving precise directions from memory versus just going there.
  - Describing coins in detail versus recognizing them.
  - Experienced computers users can only recall the names of some frequently used menu items when they are in the middle of a task and they are about to need that item. Mayes, et al. (1988)
  - Or: How do you list your full class schedule? LF or 53.

Example: Put together a list of books on Janus
- I want a list of books on "group dynamics" so I could go to the library and browse through them.
- I was prepared to just cut and paste from the UO Library on-line catalog.
- The device dictated some of the details of the task.

Seven Stages of Human Action
- A model of how people interact with the world.
- Why does failure occur in this cycle?
  - Gulf of execution:
    - Difference between intentions and allowable actions.
  - Gulf of evaluation:
    - Amount of effort needed to interpret how well intentions and expectations have been met.

Examples of Gulfs of Execution:
- You are done putting a file onto a floppy disk and want to get the disk out of the machine.
- How do you lighten and crop photos using JPEGView 3.3?
- How do you.......
  - Beware! The solution is not to put in everything! Beware of Creeping Featurism!
  - How do you print a letter you just typed?

People have fundamental capabilities and limitations
- Memory
- Learning
  - An ability to interpret knowledge in the world (sometimes called "distributed cognition"), to make sense of the world, and to figure things out.
  - In general, designers rely too heavily on these.

Memory (or Knowledge)
- Procedural vs. declarative (or conceptual)
  - Procedural knowledge is only sometimes subconscious.
  - Subconscious: riding a bike.
Conscious: making muffins.
Declarative: facts about the world.

+ Long term memory versus short term memory.
  - Differ in capacity, decay time, and the type of encoding.
  - Example: Student names.

+ Natural mappings:
  Memory in the world.
  - How do you know when you have figured out the natural mappings?
  - How did Norman figure out the “Full Natural Mappings” on p. ??

+ “Natural mappings”
  - What's the point? To some extent, “natural mappings” do not exist.
  - Norman’s Fig. 3.5 (top) burner-to-control mapping was called “natural” because it provided the greatest spatial compatibility, but this was determined through empirical testing by human factors researchers, not by intuition. Real people doing real tasks, i.e. “Turn on that burner.”
  - In another experiment, (Shinar and Acton, 1978) participants were asked which controls they thought controlled which burners, and participants most frequently chose Norman’s Fig. 3.3.
  - You can’t just think about it and get it right. You can’t just ask people what they think and get it right.

+ Constraints
  - Physical - You can’t put a square peg into a round hole.
  - Semantic - Knowledge of a situation and the world limits possible interpretations of meaning.
  - Cultural - Social norms limit possibilities.
  - Logical - Some things just makes sense. But be careful. It's not always obvious what is “logical,” “intuitive,” “natural.” How do you figure it out?
  - Are these all distinct constraints?
  - Are there others?

- Labels are okay.

- Using the Seven Stages to Ask Design Questions

+ Design concepts:
  - Visibility
  - A good conceptual model
  - Good mappings
  - Feedback
People will make mistakes

Major catastrophes
- Three Mile Island
- Chernobyl
- Plane crashes

Minor catastrophes
- Accidentally cutting power, quitting applications, closing windows, deleting files.
- Annoying, time-wasting, disrupt the goal-completion cycle, distract from task completion

All too often these problems are dismissed as "operator error."

Many errors can be avoided.

How to design for human error

Prevent them to begin with.

Design an easy-to-use system
- Keep it simple, just what is needed for the task.
- Compare an unnecessarily complex word processor to an ATM - which is more likely to lead to errors?
- Apply Norman's guidelines.
- Test with real people, real tasks.

Require confirmation for potentially destructive actions.

When possible, allow users to recover from destructive actions.
- How are these last two built into interfaces?

Require confirmation for potentially destructive actions

Avoid errors in the first place with forcing functions

Convey the right level of severity

Don't just tell the user that something is wrong, but also offer a suggestion as to how to fix it. Explain:
- What happened?
- Why did it happen?
- What can I do about it?
- (Apple Macintosh Human Interface Guidelines)
- Be sure the alert is necessary.

Be sure the alert is necessary: Do not over-warn

Advertisements, tool tips, unsolicited help alerts, and other unnecessary extra alerts will lead to more errors.

Signal detection theory has demonstrated that the harder it is to distinguish a signal (a warning that should be heeded) from noise (routine computer output, including warnings to be ignored), the more likely a person is to miss a signal.

People's responses in a signal detection task are affected by their perceptual sensitivity (d') and their response bias (β).
Possible responses in a signal detection task

What the device presents

<table>
<thead>
<tr>
<th>User's response</th>
<th>Noise alone</th>
<th>Noise plus signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes, signal</td>
<td>False alarm (error)</td>
<td>Hit</td>
</tr>
<tr>
<td>No signal</td>
<td>Correct rejection</td>
<td>Miss (error)</td>
</tr>
</tbody>
</table>

Possible responses in a signal detection task

What the device presents

<table>
<thead>
<tr>
<th>User's response</th>
<th>Noise alone</th>
<th>Noise plus signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes, signal</td>
<td>False alarm (error)</td>
<td>Hit</td>
</tr>
<tr>
<td>No signal</td>
<td>Correct rejection</td>
<td>Miss (error)</td>
</tr>
</tbody>
</table>

- **Error Recovery**
  - **Undo and Redo**
  - But undo how many steps?
  - People are going to make mistakes.
  - Build "forgiveness" into your systems.
    - "You can encourage people to explore your application by building in forgiveness. Forgiveness means that actions on the computer are generally reversible. People need to feel that they can try things without damaging the system."
    - (Apple Macintosh Human Interface Guidelines)
  - Java Swing builds support for undo and redo support right into the class hierarchy.

- **There is a risk of over-reliance on the error-correction capabilities of the system**
  - Airplane ground avoidance systems
  - Anti-lock brakes
  - Hard drives not backed up
  - Spellcheckers

- **Another way to design for error: Design for the extra-ordinary user**
  - In extra-ordinary circumstances, the ordinary user becomes the extra-ordinary user.
  - The "normal" users often suffer from these impairments when...
    - In a crowded, noisy office.
    - Driving a car.
    - Doing too many things at once.
    - Under intense pressure to meet a deadline.

- **Human error and system design**
  - Be suspicious when catastrophes are categorized as "operator error."
  - Many errors can be avoided.
  - Your users are human, have limited capabilities, and are going to make mistakes.
  - Build your systems to avoid errors and irrecoverable disasters.
The Design Challenge
- Chapters 6 & 7 of Don Norman’s The Design of Everyday Things

Common Problems with Designs
- Too many controls
- Poor feedback
- No natural mappings
- No conceptual model
- Manufacturing constraints
- Aesthetic goals

Why is good design hard?
- World and people always changing
- Predicting human behavior is not an exact science
- Complexity of design process
- Conflicting goals
- THEREFORE:
  Designs evolve in the world

What works against evolving good designs?
- Not enough time to design right
- Need for individuality in design
- Designer-centered design
  - Designers are not typical users
  - Designers’ clients may not be users
- Creeping featurism
- Worshipping complexity and technical sophistication

"Easy to Learn and Use": 7 Principles for Good Design
- Put knowledge in the world; exploit knowledge in the head
- Simplify the structure of tasks
- Make things visible
- Get the mappings right
- Exploit power of constraints, both natural and artificial
- Design for error
- When all else fails, standardize

1. Put Knowledge in the World and in the Head
   - Put knowledge in the world
     - affordance
   + Exploit knowledge in the head
     - conceptual models

2. Simplify the structure of tasks
   - Focus on the most useful functions
   - Minimize steps for tasks, especially frequent (core) ones
   - Organize functionally
   - Modularize
   + Consistency
     - Use identical actions for identical goals
     - Use identical feedback for identical state changes
     - Give user the control

3. Make things visible
   - Affordance: cues of how to use
   - Controls not visible
- Too much "visibility": Which controls do what?
- Conceptual model: Glass box versus Black box
+ Bridge the Gulf of Execution
  - options readily available
  - show mode changes
+ Bridge the Gulf of Evaluation
  - always give understandable feedback for actions

+ 4. Get the mappings right
  + Physical
    - Spatial positioning and movement of controls related to result
  + Psychological
    - Flashing draws attention to most important item
  + Logical
    - Light switch has only two states
  + Cultural
    - Searching follows scanning of reading (left to right, top to bottom for English)
    - Common icons (trashcan, mailbox) have cultural meanings

+ 5. Exploit the power of constraints
  + Constraints limit possible actions or interpretations
  + Physical
    - Example: Diskette ejection
    - Example: Button has only two states
  + Cultural
    - Example: Hot water on the left
    - Example: Logging on requires typing your full name and password

+ 6. Design for Error
  - Understand the causes of error and design to minimize
  + Make it easy to discover errors understandable
    - feedback
    - good messages
    - mapping of conceptual model to system image
  + Make it possible to correct errors
  + Make it possible to reverse actions
    - undo
  - Make it hard to do what can not be reversed
  - Test designs with users!

+ 7. When all else fails, standardize
  - Standardization is creating cultural constraints!
  - Standardization promotes consistency and thus transfer of training
+ Pitfalls
  - Lowest common denominator
  - May not be the best
  - May have to violate standard for better design

+ The Paradox of Good Design
  - If it is a good design, we frequently will not know it!
A Brief History of Building GUI's in Java - Part One

+ A Brief History of Building GUI's in Java - Part Two

+ A Brief History of Building GUI's in Java - Part Three

+ Why Swing?

+ Swing Resources
To get started with Swing, follow these steps:
- Log onto a Solaris workstation in 100 Deschutes.
- Set up your paths
  - Described in the "Java Programming Environments for this Class" web page
- Go to Sun's Swing Tutorial.
  - It's the first link on the online version of the Project 2 handout.

Click right arrow 4 times
- (Image from http://java.sun.com/docs/books/tutorial/uiswing/start/index.html
  4/12/00)
- Download and run SwingApplication.java
- Contact the Prof or GTF if you have any trouble with this.
You've now started Project 2
- Questions on Project 2?

Start Project 2 Early
- The following are not acceptable excuses for a late project:
  - "There were no workstations in 100 Deschutes available."
  - "Sun's web site crashed"
  - "You didn't respond to my email in time."

Code Example:
Printing a Vector of Strings
Using an Iterator
Vector v = new Vector();
... // Fill the vector
Iterator itor = v.iterator();
Object o = null; // used in the loop
while (itor.hasNext())
{
  o = itor.next();
  if (o instanceof String)
    System.out.println((String) o);
}

Swing GUI code does not reveal a clear thread of the flow of control
- You do not write code that you can walk through in your head from beginning to end by just looking at the code.
- Remember, GUI's are interactive.
Instead, you do the following:
- Create a top-level container, typically a frame.
- Show the frame.
- Leave it to the application to take care of itself.

The Tiniest Swing Application Possible
import javax.swing.*;
public class SwingApp {
  public static void main (String argv[]) {
    JFrame frame = new JFrame("Tiny Swing App");
    frame.show();
  }
}
Add functionality by expanding the recipe as follows:
- Create a top-level container.
- Create components and add them to the top-level container.
- Tell some components to listen for events.
- Show the frame.
- Leave it to the application to take care of itself.
  - The listeners will listen for events and, when they hear an event, will do what you told them to do.

The basic structure of a Swing Program
- Let's walk through

SwingApplication.java Walkthrough:
General Structure
+ imports at the top
  - An old-style swing import is commented out
  + No "include" statements.
  - If there were other classes and the .java files were in the directory, they would be automagically updated and included. (?)
+ One class in the file
  + The file is the exact same name as that class
  - Compile with "javac SwingApplication.java"
  - Run with java SwingApplication
  - "main" at the end
  - No clear thread and flow of control.

public interface ActionListener
  - (from Java API)
  - extends EventListener
  - The listener interface for receiving action events. The class that is interested in processing an action event implements this interface, and the object created with that class is registered with a component, using the component's addActionListener method. When the action event occurs, that object's actionPerformed method is invoked.

The imported packages
- import javax.swing.*;
- //import com.sun.java.swing.*;
- import java.awt.*;
- import java.awt.event.*;

How to add a beep for debugging:
import java.awt.Toolkit;
Toolkit.getDefaultToolkit().beep();
+ Working with Vectors
- `Vector v = new Vector();`
- `v.addElement(new Song("Jingle Bells", 80));`
- `(v.firstElement()).print();`

- Java looks for an `Object.print()`
- You must cast the `Object` as a `Song`:
- `((Song)(v.firstElement())).print();`

+ Other Java Hints
- Be sure to delete your *.class files periodically. It doesn’t always get the dependencies right.

+ Iterators
- Problem:
- You can’t access an element without moving the iterator.
- Solution: Use a `ListIterator` and follow every `next()` access with a `previous()`.

+ Events
- Change in status that can initiate a response from the computer.
- Examples:
  - Click a mouse button -> Mouse event
  - Press a key -> Keyboard event
  - Move, hide, click in title bar of window -> Window event
- For your program to respond to events
  - Define a `listener class`
  - Attach an instance of that class (`a listener`) to a component

A Listener Class
```
public class myWindowListener extends WindowAdapter {
    public void windowClosing (WindowEvent e) {
        System.exit(0); return;
    }
}
```

// `WindowAdapter implements WindowListener`

A Listener Instance
```
public class App {
    public ... main (String argv []) {
        JFrame f = new JFrame("Window");
        f.setSize(300, 100);
        f.addWindowListener(new myWindowListener());
        f.show();
    }
}
```

Another Listener Instance
```
// from Sun’s SwingApplication.java
frame.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
        System.exit(0);
    }
});
```

// Anonymous inner class
**Listeners Create "Delegation"**
- Components delegates responsibility to attached listeners.

**Benefits of listeners:**
- Refine pre-existing classes, reuse behavior.
- Can be attached to and deleted from components dynamically.
- Keep track of state, such as number of times an event occurred (as in the Sun demo).

**The Model-View-Controller (MVC) Architecture**
- **Model:** The data
- **View:** The display of the data
- **Controller:** How the UI reacts to events

**Example:** scroll bar
- **Model:** min, max, current position.
- **View:** What it looks like.
- **Controller:** Drag and move, click on the ends.

**Benefits:** Increased flexibility and reuse.
- Where are they in your Java Swing classes?

**Swing Simplifies the MVC Architecture**
- Just "Model-View" or "Component and UI Delegate" 
- **Model:** Info about the component's state 
- **UI Delegate:** How to draw the component and how to react to events.

**Example:** JTable
- **Model:** TableModel
  - JTable(data, columnNames), getModel(), getValueAt(i, j)
- **UI Delegate:** Handle all GUI responsibilities, including displaying and handling events
  - addMouseListener()

**Layout Managers**
- Organize your components within a logical hierarchy of containers.
- Nest components into subsets and manage the organization and arrangement of each subset individually.

**Layout managers are responsible for two tasks:**
- Arrange the components in a container
- Calculate the sizes of containers

**BorderLayout**
- From Sun's Java Tutorial
- Components can only be added to one of five regions.
- The components are often other containers, such as JPanels
+ Upcoming Reading Assignments
- Tues - 4/25/00 - N&L 1, 2, 4
- Thur - 4/27/00 - N&L 4 & 5
- Tues - 5/2/00 - N&L 6 & 7

+ Project #3
Interactive System Design
- In the remainder of the course you will build an interactive system
- Approaching the ideal way to design a system
- Similar to a project series you might have in a Software Engineering course, but more emphasis on the user-centered design process

+ How different from S/W Eng?
+ What you might see in a Software Engineering set of projects
  - Systems Requirements Spec.
  - System Design Specification
+ The design process from a system-building procedure
  - Detailed discussion of system design
  - More detailed system-design documents
  - System documentation
+ Object-oriented design techniques and methodologies
  - Object, control, and functional model
  - Quality assurance
  - Verification and validation
  - Change control and management
  - Project management

+ Overlap with
Software Engineering projects
+ System Design Specification
  - Problem Statement
+ Systems Requirements Spec.
  - Description of functionality
  - Object and dynamic models
  - Building a system

+ What you will probably only see here
  - User-centered problem statement
  - Emphasis on the user and the task
  - User-centered design approach
  - User and task study
  - Task analysis
  - Conceptual model or metaphor
  - Detailed usability requirements
  - Detailed user interface specification

+ One-Sentence Design
Problem Statement
+ Includes
  - Form of the solution
  - Level of support, or usability
  - Supported activity
  - Performers of action (users)
+ Example: Design a cash-operated machine for quick, easy purchase of railway tickets by passengers.
+ Why one-sentence?
+ Write a one-sentence and four-sentence design problem statement for a system of your choice.
Describe your User and Task Study
- Find a real potential user of your system
- Conduct an interview or observational study
- For example: You want to build the first Palm Pilot, so you observe people as they accomplish real and specific tasks using their current paper-based planner and address book.
- First just describe the study, including sample forms.

Present the results from the User and Task Study
- Present the results of the study
  - First describe what you learned about the participants in your study
    - The reader can determine if this is a representative user
  - Then describe what you learned about the tasks
    - Use an activity or task model, such as a hierarchic model
      - Example: Hierarchic model for making a lunch date.
    - The language should be that of people doing tasks, and nothing about the device.

Functionality Overview
- Now you are talking in the language of the computer.
  - First a high-level summary, and then a more detailed description.
  - The kind of thing you would read in the review of a software product.
  - Prioritize the functionality. You will not be able to implement all of it.

Conceptual Model or Metaphor
- What is the User Model you expect your user to have?
  - You will probably get it wrong, but at least you have now stated it and can potentially compare it to the real user's model later.
  - Example: The user's model would be that of flipping through a paper-based address book.

Object model
- Describes the static structure of the objects in a system and their relationships.
  - Contains object diagrams, graphs in which nodes are classes and arcs are relationships among classes.

Dynamic model
- Describes the aspects of a system that change over time and specifies the control aspects of a system.
  - Contains state diagrams, graphs in which nodes are states and arcs are transitions between states caused by events.

Usability Requirements
- Learnability and Usability
  - Set objective criteria
    - Specific tasks and times
  - Core versus advanced features
  - Try to be realistic

User Interface Specification
- The dynamic blueprint
  - What the user will see
    - What actions can be performed at each screen
    - What happens as a result of each action
  - How do these relate to Norman?
  - How do these relate to Swing?
  - It all comes together
Interactive System Design: Identifying the problem and designing the solution

- Design is as much about communication as it is about creativity

Upcoming Reading Assignments
- Thur - 4/27/00 - N&L 5
- Tues - 5/2/00 - N&L 6&7

Creativity in Design
- Engineering is about creativity and innovation.
- Engineering methodologies ease the laborious aspects of design.
  + Analyzing performance
  - Hardware benchmarks, objective usability goals
  + Testing for safety and ease of use
  - Automobile crash tests
  - Integrating components
  - Tracking down obscure design knowledge.
- These are the skills of a good usability engineer

Innovation requires broader range of tools and methods
- Las casas colgadas en la ciudad de Cuenca.
- For studying unfamiliar application domains and problems and making sense of what you find
- For describing novel solutions to design problems
- For testing new designs, even when the use is not clear
- Researching to fill a gap in available engineering knowledge
- When to use one method versus another

The creative engineering design process
- Analyze and Synthesize
  - Gain an understanding of the problem.
  - Address the problem with the appropriate available methods.
- Why do we talk about the process so much?
  - These are no longer textbook homework assignments.
  - To slow you down and make you think about the problem

Design activities that contribute to the basic iterative design cycle of analysis and synthesis
- How are these similar to the loop you already know so well:
  - Design/re-design
  - Program
  - Test
  - Debug

Defining the Problem
- You need a clear problem statement
  - To communicate your goals
  - To track your progress towards your original goal
  - To solve the right problem

The components of a problem statement
- The human activity that the system will support
- The users
- The levels of support (usability) that the system will provide
- The basic form of the solution
- There should be a causal link from a situation of concern to the problem statement.
- Each component evolves into a major area of design work.

+ Elaborating a One-Sentence Design Statement

+ The human activity and the users
  - Collect data using user studies: interviews, observational studies (verbal protocol or dialogue), or questionnaires.
  - Analyze the data to build user and task models.
  - For an educational piece of software, who are your users?

+ Level of support
  - Identify the critical aspects of system performance. There will always be trade-off's.
  - Establish detailed, complex, objective criteria of evaluation.

+ Form of the solution
  - Ultimately, this will be your entire design specification.

+ The Activity (or Goal) to Be Supported

- Let's hear the proposed activities.
- Goal-oriented activities are broken down into tasks.

+ A task:
  - A specific individual piece of work
  - Something you could tell your personal assistant and it would get done.
    - For example: “Fetch me a cup of coffee.”
  + Can be broken down into subtasks or steps
    - Goal-oriented behavior tends to be hierarchical.
    - Added twist: People choose the methods they use to accomplish a task.

+ Example Task

+ Read the paper co-authored by Alistair Sutcliffe that will be discussed in the HCI meeting today.
  - Find about an hour of time.
  - Get a cup of coffee.
  - Get the paper.
+ Read the paper.
  - Figure out the number of pages.
  - Read the title and abstract.
  - Flip through all the pages reading the headings and looking at the pictures.
  - Read the first sentence of every paragraph.
  - ...

+ In-class activity

- Think about the activity of “Getting through the day today.”
- Identify a task that you will accomplish today and break it down into one level of subtasks.
- Remember: A task is a specific individual piece of work, something that you could hand off to your personal assistant.

+ The User

+ General human performance capabilities
  - Discussed somewhat in Chapter 3, we’ll read it later
+ The target user will usually also have uniquely-definable characteristics.
+ What are the characteristics of:
  - An Internet-user?
  - A UofO student?
  - A UofO faculty?
  - A first-grader in a Brooklyn public school?

+ Level of Support:

Usability Factors
+ It's more than just "quick, easy"
  - Speed of performance
- Incidence of errors
- Ability to recover from errors
- Ease of learning: how much to learn and how long it takes
- Retention of learned skills
+ User's ability to customize the system to suit their way of working or the situation of use
- Screen colors in a word processor
+ Ease with which people can re-organize their activities supported by the system
- Tektronix order-entry system
- User satisfaction

+ Usability targets
- Not all are quantifiable.
- There will probably be some correlation between quantifiable factors (i.e. time and errors) and non-quantifiable factors (i.e. user annoyance).
- My dream: A standardized EPA miles-per-gallon usability rating on all shrink wrapped software, a rating that incorporates a spectrum of usability factors.

+ The form of the solution
+ Hardware platform and OS
- Dictated to you in this class.
- But the problems you identify might easily be better solved with others.
- And sometimes a computer is not needed at all.
+ The User Interface
- Don't try to design a complete solution on the spot.
- Don't just start drawing screenshots.
- Identify different potential forms.

+ Next topics
+ What follows after problem definition?
- Design (Ch. 4)
+ A critical component of the design process:
- User Studies (Ch. 5)
Interactive System Design: Identifying the problem and designing the solution

You've identified the problem. Now you must design a user-centered system and communicate it to others:
- During the design process.
- As a product of the design process.

Topics for today:
- User Studies
- Design Notations

Announcements

Upcoming Reading Assignments
- Tues - 5/2/00 - N&L 6&7
Teaching Effectiveness Program visitors
- Georgeanne Cooper next Tuesday
- Videotaping next Thursday

What follows after problem definition?

- Design
  - Design is as much about communication as it is about creativity

User Studies

- Interviews
- Observations
- Questionnaires
- For All User Studies
  - You must find representative users.
  - Design meetings at Tektronix

Interviews

- Why they are good
  - Social
  - Can instantly follow leads that appear
  - You can pursue areas of interest
- Potential problems
  - Some people are not great interviewers
  - DiCaprio interviewing Clinton
  - People might not open up
- Notations
  - Textual summaries
  - Statistical analyses
  - Not aware of any special notations

How to conduct a good interview, Part 1

- Listen -- the most important part
  - Listen more than you talk
  - Listen actively
  - Assume the user knows a lot about his or her work
  - Treat the participant/user as a partner, not a research subject.
  - You are examining the activity, not the user.

How to conduct a good interview, Part 2

- Plan the interview. Know what you are trying to learn
Use different kinds of questions.
- Closed versus open
- General versus specific
- Factual versus hypothetical

Ask neutral questions
- Remember: User's tend to blame themselves, and people
  - How did it go" vs. "Hard, right?"
  - "Do you have a user's manual" versus "Why didn't you just look it up?"

How to conduct a good interview, Part 3
- Respect silence
- Be conscious of body language and other signals
- Be sensitive to cultural differences.
- Keep the conversation going.
- Probe to get additional information
  - Brings it more alive for team members who are not there
  - Good systems incorporate the users vocabulary

Observational Studies
- Potentially the most "natural" task setting.
  - Video recording is a good tool
    - Typically need to get permission
    - Frame the computer screen or the whole room?
    - Transcribe using verbal and physical protocols (set of conventions or codes)
    - Transcription time
  - Ethnographic field studies
    - From anthropology
    - In vogue a few years ago
  - Potential problems
    - Hawthorne effect

Tips for Conducting an Observational Study, Part 1
- Get to the most realistic task situation possible.
  - Make it a cooperative venture.
    - Work to build rapport.
    - You are interested in what users do, however they do it.
    - Make it clear you are not testing the person, but studying the work environment supporting them.
    - Don’t let the user make you the “expert.”
  - Be flexible about intervening tasks. These also might be part of the data that you want.

Tips for Conducting an Observational Study, Part 2
- Mind the same rules as in the interview, but talk even less. Stay friendly but neutral.
- Mostly ask clarifying questions.
  - Observe the user’s environment.
    - What kinds of “knowledge in the world” do they use? Lots of Post-It’s?
    - What are the visual and sound characteristics? Wide open office? Factory floor?
    - What kinds of interruptions occur?

Tips for Conducting an Observational Study, Part 3
- Try to get to the user’s goals. It’s not just “what are they doing now”, but "what is the higher level goal in the hierarchy?"
  - Task --> Goal
    - Change the channel --> watch the debate.
    - Turn on the oven --> feed the guests.
    - Read the notes from the last customer visit --> secure a sale.
    - Install the modem --> download assignments from home.
Tips for Conducting an Observational Study, Part 4

To get out the goals
- Ask them what they are doing now. They will probably say tasks or secondary goals. Then ask them “Why do you want to do that?”
- Note what triggers the task, where the user starts it, and where it ends.

Get them to talk
- Ask users to “think aloud”
- Use co-discovery
- Take lots and lots of notes. Write down everything. This is your only chance. You will not regret it.

Questionnaires
- Relatively easy to distribute
- Difficult to design them well
- No immediate social buy-in
+ Response rates not always good
  - 1990 Census - 65%
  - 2000 Census - just hit 65%
+ Questionnaires via email might be considered a spam.
  - If you must, then tightly constrain your userlist, use bcc:, keep it short.

Design Processes and Representations
+ Different styles of notation for different steps in the design process
  + For conveying the process of design
    - N&L’s design representation and process flow diagram.
  + For conveying the specification requirements
    - Outline document
  + For conveying the software design
    - Object, Dynamic, and Functional Models

Design Processes and Representations
+ For analyzing user data
  - Dialogue or interview fragments
  - Questionnaire summaries
  - Hierarchical task decomposition
  - Process flow diagrams
+ For conveying the UI design
  - Interaction state transition networks
  - Verbal scenarios
  - Storyboard

Models in Rumbaugh’s OMT
+ Object model
  - Describes what changes—the static structure of the objects and their relationships.
  - Contains object diagrams, graphs in which nodes are classes and arcs are relationships among classes.
+ Dynamic model
  - Describes when it changes—the aspects that change over time, the control specifications.
  - Contains state diagrams, graphs in which nodes are states and arcs are transitions between states caused by events.
+ Functional model
  - Describes how it changes—the data value transformations within a system.
  - Contains data flow diagrams, graphs in which nodes are processes and arcs are data flows. A data flow diagram represents a computation.

Conclusion
- You’ve identified the problem. Now you must design a user-centered system and communicate your design to others.
Topics for today:
- Systems Analysis and Design
- Requirements Specifications

Announcements
- Your progress
- Upcoming Reading Assignments
  - Thursday - 5/4/00 - N&L 3 (optionally 8.4.2)
  - Tuesday - 5/9/00 - N&L 8
- Teaching Effectiveness Program visitors
  - Georgeanne Cooper
  - Videotaping Thursday

Modals in Rumbaugh's OMT
+ Object model
  - Describes what changes--the static structure of the objects and their relationships.
  - Contains object diagrams, graphs in which nodes are classes and arcs are relationships among classes.
  - Analogous to Entity Relationship Diagrams

+ Dynamic model
  - Describes when it changes--the aspects that change over time, the control specifications.
  - Contains state diagrams, graphs in which nodes are states and arcs are transitions between states caused by events.

+ Functional model
  - Describes how it changes--the data value transformations within a system.
  - Contains data flow diagrams, graphs in which nodes are processes and arcs are data flows. A data flow diagram represents a computation.

Remaining Topics for Today
+ Hierarchical Task Models
  - For prediction and design
- Interaction State Transition Network versus OMT Dynamic Model
- Examples of UI Specifications
- Requirements Specifications for Palm Pilot purchase
+ Announcements
  + Upcoming Reading Assignments
    - Tuesday - 5/9/00 - N&L 8 (discussion on GOMS)
    - Thursday - 5/11/00 - Cognitive Walkthrough (possibly brief downloadable reading)
    - Tuesday - 5/16/00 - Nancy Cheng from the Dept. of Architecture
    - Bill Holmstrom from the Teaching Effectiveness Program will be videotaping
      today.
    - Remind me to take a break.

+ Good quiz questions for Chapter 8 on Tuesday:
  - What does KLM stand for?
  - Name the three usability and analysis methods discussed in the chapter.
  - The first two methods discussed require that the user's ______ be clearly
    defined.

+ Topics for today:
  - Fundamental human performance capabilities
  - OR
  - The Human Virtual Machine

+ Basic Human Processes
  + N&L call it the Human "Virtual Machine"
    - Perceptual Processors
    - Cognitive Processors (Decision)
    - Motor Processors (Action)
    - Memory
  + Other versions
    - The Model Human Processor
    - EPIC: Executive Process-Interactive Control

+ Memory
  + Working Memory
    - Fast to store
    - Fast to retrieve
    - Fast to decay
    - Small
  + Long Term Memory
    - Slow to store
    - Varying speeds to retrieve
    - Slow to decay
    - Enormous

+ What do the processors have in common, and how do they differ?
  - All require processing time.
  - Each processor can operate to some extent in parallel with the others, but...
  - There will be capacity limitations and bottlenecks within
    - Each perceptual subsystem
    - Each limb subsystem
    - But not in the central cognitive processor
  - There is no evidence for a central cognitive processing bottleneck.

+ Perception
  - After the physical stimulus hits the nerve endings, roughly 100 to 200 msec is
    required for the perceptual information to be encoded and to arrive at the
    cognitive processor.
  - Visual acuity is best at the center of the gaze and gradually deteriorates as
    items get further from the center of the gaze.
- The foveal region, roughly 2° of visual angle across, has the highest acuity.
- You must get an object into your fovea to discern its detail.
- How do you get the light rays from physical objects into your fovea?

**Action**
- Manual motor activity
  - Can be predicted by Fitts' law, but Fitts' law does not explain the subprocesses involved.
- Ocular motor activity
  - Most eye movements are saccades (as opposed to smooth pursuit).
  - Saccades are very fast (roughly 5 to 20 msec).
  - But intersaccade latencies are much longer (typically a minimum of roughly 225 msec).

**Decision**
- This is where the action is—the tough one to predict.
- Hick's Law
  - Decision time increases as a logarithmic function of the number of possible choices.
  - But does not explain the subprocesses involved.
- Human cognitive strategies are complex.
- Depending on your level of analysis, single decision times can range from milliseconds to minutes.
- How do you even begin to figure out the strategies people use?

**How do you begin your analysis of the strategies people are likely to use to accomplish a task?**

**Task analysis**
- Figure out the piece of work to be done
- Decompose the task into its subcomponents
- Start thinking about how a person would apply their "virtual machine" to accomplish the task.

**Put the Virtual Machine to work**
- Predict task completion times
  - KLM and GOMS incorporate performance assumptions from the Human Virtual Machine.
- Use it when you think about design
  - Decompose the task into its component processes.
  - Think about how people would apply their machinery to accomplish the task.
  - Simplify the decisions to be made.
  - Remove the perceptual and motor bottlenecks.
  - Maximize the throughput of information and efficient task execution.

**The Future of the Human Virtual Machine**
- Interface designers tools will use tools that automatically predict aspects of human performance, such as visual search times.
- These tools will be built by incorporating what is known about the human virtual machine.

**The Problem: How to Predict Visual Search Performance**
- Example: How long will it take someone to find "Computer Accounts?" How will they do it?
+ Announcements
  + Upcoming Reading Assignments
    - Thursday - 5/11/00 - Cognitive Walkthrough
    - Tuesday - 5/16/00 - Interface presentations, with Prof. Nancy Cheng from Architecture
    - Thursday - 5/18/00 - Interface presentations
    - Then: User testing.

+ Interface Presentations
  - Prepare a presentation of your user interface design, including overhead transparencies that will allow us to walk through your interface.
  + Presentation must include:
    + Introduction (one transparency)
      - One-sentence problem statement
      - Describe the users
      - Specific sample tasks
    + One transparency per interface screen
    + All groups must be ready to present on 5/16/00.
    + 5% of Project 4 grade

+ Topics for today:
  - Fitts’ law
  + Predictive Engineering Models
    - KLM
    - GOMS

+ Fitts’ Law

\[
T = K \cdot \log_2 \left( \frac{\text{Distance}}{\text{Width}} + 1 \right) \text{ms}
\]

- (N&L use Mackenzie’s version. There are other versions that differ slightly.)
- Very accurately predicts relative differences pointing times as the distance and the width change. (K cancels out)
- Predicts pointing times reasonably well, with K set around 100.
  - A constant coefficient of about 100 should also be included.
  - Assumes target is visible and no visual search is necessary.

+ Predictive Engineering Models
  + Allow the analyst to predict user performance based on
    - A specification of the design
    - Task analysis
    - Device description
    - Fundamental laws of performance
  - All the other engineering disciplines predict system performance in this manner. Usability analysis should be able to predict interface usability in the same manner.
  + Assumptions
    - Well-defined task
    - Practiced user

+ How to Build Usable Systems
Choose Tasks

Specify Design

Build Prototype

Test Usability of Prototype

Find Problems? Yes

No

Build System

- Problems with this approach:
  - Slow, expensive, and does not necessarily explain why a design is better
  - Solution: Incorporate predictive modeling

+ Predictive Models
  - Predictive Engineering Model:

- Physical test:


+ Why engineering models are good
  - Faster, cheaper, explain why a design is better
  - Can contribute to a design specification

+ Incorporating Engineering Models Into System Design
Choose Tasks
   Specify Design
      Build GOMS Model
         Test Usability with GOMS Model
            Find Problems? Yes
               No
               Build Prototype
                  Test Usability of Prototype
                     Find Problems? Yes
                        No
                        Build System

Human Factors

+ Predictive Engineering Models in Usability Analysis
  + GOMS Models
     - KLM
     - NGOMSL
     - (others)

+ By the way, on page 173:
  - Add "Press or release mouse button: 0.1 sec"
  - Use the 0.28 for your typing predictions.

- The remaining text is taken directly or adapted from "Cognitive Analysis of Dynamic Performance: Cognitive process analysis and modeling" by Boehm-Davis, Gray, John, and Kieras, materials presented as part of the Free GOMS Tutorial, Pittsburgh, PA, May 17, 1999. Available at http://hfac.gmu.edu/Free_GOMS/FreeGOMS.ppt.

+ GOMS as Predictive Modeling
  - GOMS analysis produces a model of behavior
  - Given a task, the model predicts the methods, or sequences of operators, that a person will perform to accomplish that task
  - Can look at the GOMS model in different ways to qualitatively and quantitatively assess different types of performance

  (From Boehm-Davis, Gray, John, and Kieras, 1999)

+ Scope of GOMS:
  What it can do
  - Predict the sequence of operators an expert will perform
  - Predict performance time of expert users - even in real-world situations
  - Predict learning time in relatively simple domains
  - Predict savings due to previous learning
  - Help design on-line help and manuals

  (From Boehm-Davis, Gray, John, and Kieras, 1999)
+ Scope of GOMS:
What it can do (continued)
+ GOMS has been applied to both:
  - User-driven interaction
  - "Situated" or event-driven interaction

(From Boehm-Davis, Gray, John, and Kieras, 1999)

+ Scope of GOMS:
What it cannot do
- Predict problem-solving behavior
- Predict how GOMS structure grows from user experience
- Predict behavior of casual users, individual differences...
- Predict the effects of fatigue, user preference, organizational impact...

(From Boehm-Davis, Gray, John, and Kieras, 1999)

- Insert N&L Overhead 8-12 here. Also 6-9, 10, 11.

+ NGOMSL
  - Natural Language GOMS
  - Based on structured natural language notation and a procedure for constructing them
  - Models are in program form
  - Adds a control structure to KLM: Hierarchical goal stack

(From Boehm-Davis, Gray, John, and Kieras, 1999)

+ NGOMSL - why?
  - More powerful than KLM. Much more useful for analyzing large systems
  - More built-in cognitive theory
  - Provides predictions of operator sequence, execution time, and time to learn the methods
  - Represents the user's strategy.

(Adapted from Boehm-Davis, Gray, John, and Kieras, 1999)

+ Example of NGOMSL
  + Carry out a GOMS analysis of the following task involving a digital clock:
    - Set the clock
    - Top level goal: SET CLOCK

(Adapted from Boehm-Davis, Gray, John, and Kieras, 1999)

+ Example of NGOMSL: Goals
  - Goals and subgoals

(Adapted from Boehm-Davis, Gray, John, and Kieras, 1999)
+ Example of NGOMSL: Operators
  Operators are the most elementary steps in which you choose to analyze the task.
  - Reach <type> button
  - Hold <type> button
  - Release <type> button
  - ClickOn <type> button
  - Decide: if <x> then <y>
  - Verify

(Adapted from Boehm-Davis, Gray, John, and Kieras, 1999)

+ Example of NGOMSL Methods
  + Top-level user goals
    - SET-CLOCK
  + Method for goal: SET-CLOCK
    - Step 1. Hold TIME button
    - Step 2. Accomplish goal: SET-HOUR
    - Step 3. Accomplish goal: SET-MIN
    - Step 4. Release TIME button
    - Step 5. Return with goal accomplished
  + Method for goal: SET-<digit>
    - Step 1. ClickOn <digit> button
    - Step 2. Decide: If target <digit> = current <digit>, then return with goal accomplished
    - Step 3. Goto 1

(Adapted from Boehm-Davis, Gray, John, and Kieras, 1999)

+ Example of NGOMSL: Selection rules
  - Change the SET-<digit> method into a selection rule that calls one of two possible methods:
    - Selection rule for goal: SET-HOUR
      - If target HOUR ≤ 4 hours from current HOUR, then Accomplish Goal: ClickOn HOUR
      - If target HOUR > 4 hours from current HOUR, then Accomplish Goal: Click&Hold HOUR

(Adapted from Boehm-Davis, Gray, John, and Kieras, 1999)

+ NGOMSL - Overall Approach
  - Step 1: Perform goal/subgoal decomposition
  - Step 2: Develop a method to accomplish each goal
    - List the actions/steps the user has to do goal (at as general and high-level as possible for the current level of analysis)
    - Identify similar methods/collapse where appropriate
  - Step 3: Add flow of control (decides)
  - Step 4: Add verifies
  - Step 5: Add perceptuals, etc.
  - Step 6: Add mentals for retrieves, forgets, recalls
  - Step 7: Add times for each step
  - Step 8: Calculate total time

(Adapted from Boehm-Davis, Gray, John, and Kieras, 1999)


Announcements

Next Week: Interface Presentations
- Everyone must be ready on Tuesday 5/16/00
- We’ll start on Tuesday, and finish on Thursday.
- 5% of Project 4 grade

Reading Assignment for the following Tuesday 5/23/00
- N&L 9.4 to 9.9 (but skip 9.4.3 and 9.4.4)
- *Ten Steps for Conducting a User Observation* by Apple Computer, on course web page entitled “Interesting Links”

Topic for today:
- Cognitive Walkthrough

Cognitive Walkthrough
- Focuses on one attribute of usability: ease of learning
- But other aspects of usability, like functionality and ease of use, are correlated with ease of learning
- Will push your design in the direction of ease of use.
- Not ideal for highly trained users or efficiency-oriented interfaces.
  - Nuclear power plant operators
  - Air traffic controllers
  - Telephone operators
- Assumes that users will not read the manual first
- Relates to Norman’s seven stages of human action

The seven stages of human action (Norman, 1988)

**Goals**

**Intention to act** to achieve the goal

**Evaluation of interpretations** with respect to what we expected to happen

**Sequence of actions** that we plan to do

**Interpreting the perception** according to our expectations

**Physical execution of the sequence**

**Perceiving the state of the world**

The World


Two Phases of Cog. Walk.

Preparatory
- Analysts agree on the input conditions and prepare the materials.
+ Analysis
- Analysts work through each action of every task being analyzed.

+ Preparation Phase
+ Who are the users of the system?
- The more precise the description, the better.
- Example: Users who already know how to prepare and print a simple document using Word 98.
+ What are the tasks?
- A reasonable representative collection of routine tasks.
- For existing system, perhaps just the known problem areas.
- Should relate the the problem statement.
- Should be concrete and realistic.

+ Preparation Phase (continued)
+ What is the correct action sequence for each task?
- At the level of detail necessary to guide the user through the interface.
- What would appear in a help system or manual.
+ How is the interface defined?
- Describe the prompts preceding every user action.
- Describe the interface's reaction to the actions.
- The interface does not need to be implemented yet.

+ Question
- Is the cognitive walkthrough limited to interfaces you can see?

+ Analysis Phase
- Walk through the interaction telling a credible story.
+ At every step or prompt, consider:
  + Will the user know the correct subgoal or subtask?
    - Example: Print or select printer first?
  + Will the user know that the correct action is available?
    - Example: Any clues for how to print?
  + Will the user associate the correct action with the subgoal?
    - Example: Type "lp" or find in menu?
  + If the correct action is performed, will the user know that progress is being made toward the goal?
    - Example: Is it being printed?

+ Which interfaces succeed and which fail?
- To succeed, a credible story must be told for all four questions at every step of the way.
- If a credible story cannot be told for just one question at one step, you have identified an interface failure.

+ Analysis Phase
- Good idea to videotape it, to go back and verify or retrace comments or decisions.
- As you go, record assumptions about what user would know prior to performing the task, and what the user would learn while performing the task.
- If you identify a problem, write it down, assume the correct action was made, and press on.

+ How to fix the breakdowns:
If the user does not know...
+ Which subgoal to accomplish
  - Eliminate the required action
  - Prompt the user to make the action
  - Re-organize the interface to more closely support the users' anticipated task hierarchy.
+ The action is available
  - Make the controls more obvious, as with a prompt or a menu.
+ The action is appropriate
  - Provide labels and descriptions for actions that incorporate the users’ vocabulary
  - Reword labels selected in error
+ Progress is being made
  - Prompt for the next correct action.
  - Provide feedback regarding what happened, ideally in the users’ vocabulary.

+ Let’s do a simple one
+ Preparation
  + Who are the users?
    - New user with data already loaded, or an experienced user who is not fully focused on this task.
  + What tasks will be analyzed?
    - Find Jeffrey Stolet’s phone number
    - Find Mark Timoney in New York’s work number
  + What is the correct action sequence for each task?
  + How is the interface defined?

+ Cognitive Walkthrough
  - A formalized methodology to check for gulfs of execution and evaluation
  - Focuses on ease-of-learning, but other aspects of usability are correlated with ease of learning
  - Does not assume highly trained users
  - Does not predict execution times

+ A lengthier discussion
  - is available in Chapter 4 of the downloadable book:
Announcements
- Final Reading Assignment, for Thursday 5/25/00
- Quiz questions:
  - What was the task?
  - What modeling technique was used to evaluate the interface?
  - True or False? The models demonstrated that the new interface would be faster before the interface was tested with real users.
  - Why was one interface faster than the other?

Topic for Today:
- Usability Testing

Ten Steps for Conducting a User Observation
- Introduce yourself and explain the purpose of the study.
- Tell the participant that it is OK to quit at any time.
- Tell them about the equipment.
- Explain how to think aloud.
- Explain that you will not help.
- Describe what the participant will be doing.
- Ask for any questions.
- Be vigilant, and remind the participant to think aloud.
- Debrief the participant.
- Use the data.


Key Points
- Think aloud can be facilitated by two users, "co-discovery."

The roles
- Test monitor
- Users or "participants"
- Observers

The task
- Using the keyboard, type the following as a To-Do item:
  - The quick brown fox jumped over the two lazy dogs. This is a great usability evaluation!
- There should be no typos.

Some of the questions the evaluation will answer:
- What was the typing speed for each keyboard?
- Which keyboard is easier to figure out how to set up?
- Does the leather cover get in the way?
+ Announcements
- Next Tuesday - Review for Final
- Next Thursday - Demos in 100 Deschutes

+ Topic for Today:
- N&L Case Study A: CPM-GOMS analysis and user testing of a telephone operator's workstation.
- Operational sequence diagrams

+ The major points of the study (as they apply to this class)
- A real-world case study of design, implementation, and evaluation
- Fancy GUI's are not always better than simple text interfaces.
- Usability analysis methods can predict usability issues before you test your systems with real users.
- Usability analysis methods are scientifically proven.

+ The users and the task
+ Users
  - Toll and Assistance Operators (TAO's)
  - Highly practiced at this routine activity
+ Task
  - Answer the phone call
  + Determine...
    - Who should pay for the call
    - What is the correct billing rate
    - When the connection is complete enough to hang up
  - Involves discussion with customer, typing into workstation, reading information on the workstation screen

+ The study
+ Two threads conducted in parallel:
  + Observational
    - Prototype of proposed system built, installed, and used by 24 TAO's for four months.
    - 24 TAO's continued using current system
  + CPM-GOMS analysis

+ What is CPM-GOMS analysis?
- CPM = Critical Path Method.
- Critical Path: The tasks in a project such that, if any are delayed, the whole project will be delayed.
- Similar to, but more complex, than KLM.
- Models keystrokes, but also simulates the parallelism that exists in the human information processor
- Maintains a separate thread for each human perceptual, cognitive and motor processor.
- Predicts task execution time by finding the critical path

+ Why GOMS analysis is appropriate for this task and this interface
+ Time is the important metric
  - 1 second = $3 million
+ The task is easily modeled using GOMS
  - Operators: Listening, talking, reading, keying, writing, and various cognitive activities.
CPM-GOMS is particularly appropriate because many of these subtasks are performed simultaneously. Well-practiced, expert performance. A large-scale field study was conducted concurrently. An alternative prediction (based on intuition) could be tested.

Built CPM-GOMS Models
- Built fifteen models for each workstation (for the fifteen benchmark tasks)
- For each model: Decomposed the task, established the dependencies
  + Assigned operator times
    - Some were taken from the literature (psychological studies, previous GOMS modeling)
    - Other operator times were established by observing the current workstation.
    - The proposed workstation was not observed during the modeling.

Results
  + CPM-GOMS Model:
    - Proposed workstation: 0.63 msec (4%) slower
  + Observed:
    - Proposed workstation: 0.65 msec (4%) slower
    - The models consistently underpredicted observed times by a fraction of a second, but correctly captured the difference between the two workstations.

Parallels to other interface problems
  + The execution time problems identified in project Ernestine are analogous to general usability problems in other interfaces.
  - Execution time is correlated to overall ease-of-use.

Take home messages
  + Use analytic methods early in the process
    - KLM or other GOMS methodologies
    - Cognitive Walkthrough
    - Prototype
    - Test with real users doing real tasks
    - KLM can be used effectively in the design stage.

Why the difference?
  - No benefit from new keyboard layout
  - No benefit from new screen layout
  - Keying procedures not on critical path

Miscellaneous
  - Car crash simulations
  - Incorporating pre-recorded bits of spoken text.
  - Text-only is not so bad