CISC 836
Design of Usable Computing Systems

T.C. Nicholas Graham
http://stl.cs.queensu.ca/~graham/cisc836

Examples of Usability Problems

- System is poorly documented
- System is hard to use
- System is hard to learn
- System doesn't match needs of users
- System is buggy or poorly supported
- …etc.

System Hard to Learn

- Telephone on my desk
  - E.g., buttons for "Ring Again", "Redial"
  - Voice mail
    - listen to message = 2
    - delete message = 76
    - save message = ???
    - incomplete prompts

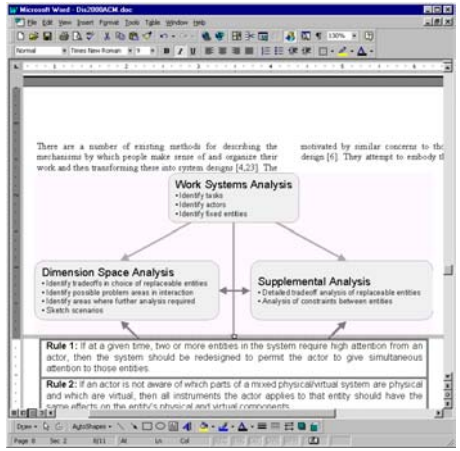
System Hard to Use

- Web Banking System
  - Different screens for
    - Balance enquiry
    - Transfer funds
    - Pay bills
    - Short, long term history
  - E.g., paying visa requires navigating > 10 screens

System Untrusted

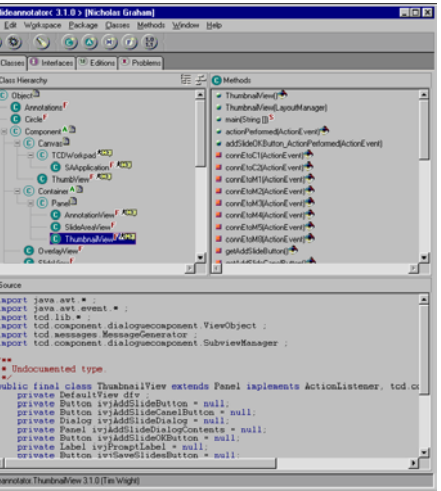
- E-bay online auction
  - Worry about safety of credit card use
  - Worry about honesty of vendors

System is Buggy, Poorly-Supported



- Positioning pictures in MS-Word
  - Positioning sometimes random, hard to tune
  - Updating cross-references repositions figures!

System Doesn't Match Needs of Users



- IBM VisualAge for Java in University Environment
  - Each machine can support only one user workspace
  - Incompatible with "walk up" terminal use in student labs

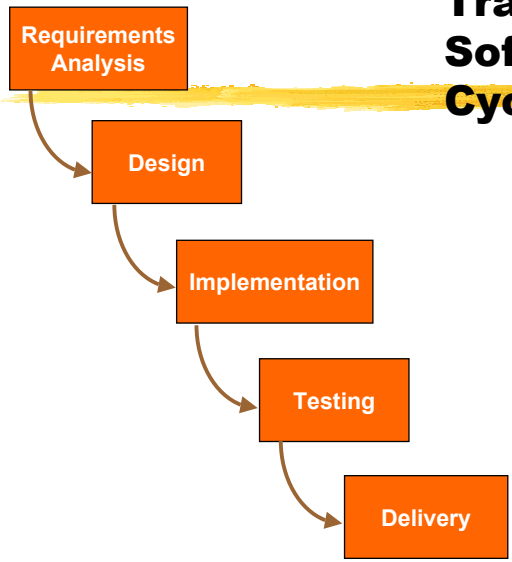
Development of Interactive Systems

- 1960's: Monolithic mainframes, cards
  - Expensive, batch, select user group
- 1970's: Glass teletypes, mini-computers
  - Cheaper, interactive, somewhat select
- 1980's: Bitmapped displays, novel I/O
  - Cheap, highly interactive, available
- 1990's: GUI's, networking
  - Extremely cheap, connected, ubiquitous
- 2000's: Information appliances, wireless
  - Extremely cheap, connected, mobile

Question for System Developers

- Can we develop systems that
  - fit the organization?
  - fit the user?
  - still meet time, budget constraints of project?

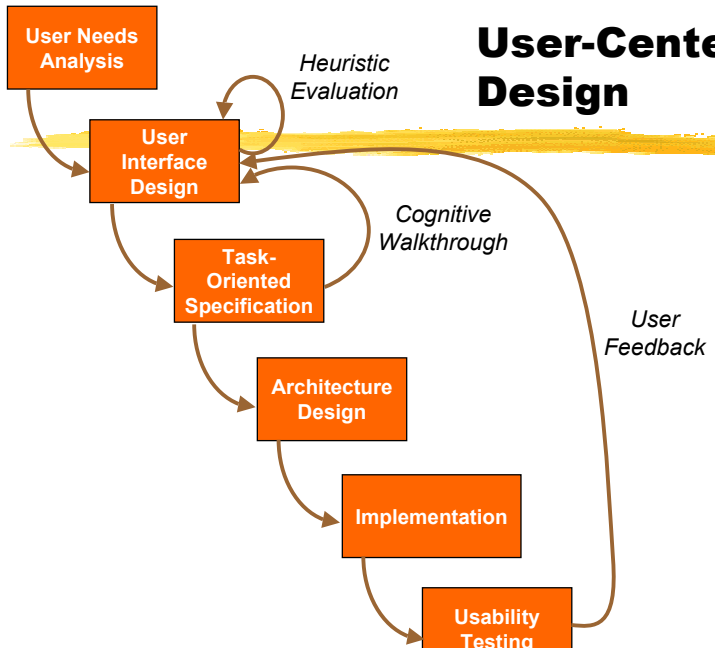
Traditional Software Life Cycle



Problems with Traditional Life Cycle

- Usability problems caught late
  - Expensive to repair when found
  - Developers become attached to designs
- If project running late, usability may be sacrificed

User-Centered Design



Goal of User-Centered Design

- Involvement and consideration of target users throughout process
- Continuous evaluation of design
- Detection of problems early

User Needs Analysis

- User characterization
- Modeling user activities, user tasks
- Modeling context of use

User Interface Design

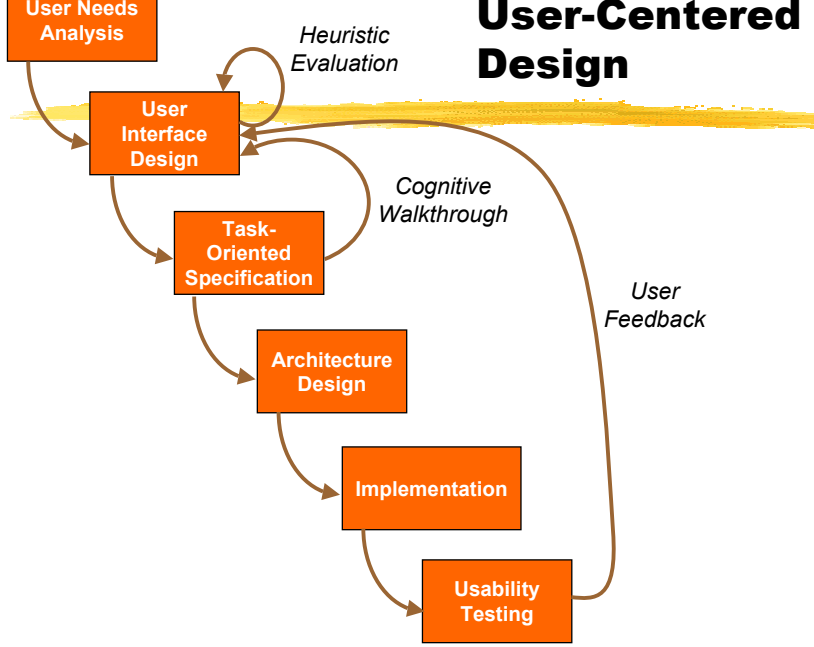
- Creative process
  - Brainstorming, metaphor, scenarios, prototyping
- User participation
- Evaluation through heuristic analysis, user feedback

Task-Oriented Specification

- Description of how user tasks carried out with system
- Extension of task model
- Bridges gap between design and implementation
- Evaluation as cognitive walkthrough

Usability Testing

- Testing with members of user group
- Can be done with prototypes or real system



User Needs Analysis

- Modeling
  - User
  - User activities, tasks
  - Context of use

User Characterization

- People
  - Age, gender, cultural characteristics
    - E.g. Radhakrishnan's field medical systems for India - can't assume literacy
    - E.g., boys on average respond to violent video game metaphors better than girls
    - E.g., checkboxes have different meaning in Chinese
    - E.g., red=danger, red=stop not universal
    - E.g., "thumbs up" sign not good in Italy

User Characterization

- Job characteristics
  - role description
  - reward structure
    - e.g., does working more efficiently lead to bonuses or layoffs?
  - turnover rate
- User background
  - training, education, skills
- Usage constraints
  - voluntary vs mandatory
  - motivators/demotivators

Task Analysis

- In context of his/her job/activity, what does person do?
- User task model
  - Generic tasks performed by user independent of any implementation
- System task model
  - Tasks supported by some system
- Stages
  - Activity analysis
  - Informal task analysis
  - Hierarchical task analysis (HTA)

Context of Use

- Physical environment
  - Noise level, lighting level, mobility requirements, encumbrances on user
  - E.g., system allowing police to create traffic tickets is used in noisy, mobile environments where police officer is on foot

Task Modeling

Discovering User Tasks

- Activity analysis, ethnographic study
  - Observing, interviewing people in work context
- Informal Task Model
  - Brainstorming
- Hierarchical Task Model
  - Structured model showing
    - task decomposition
    - temporal relations between tasks

Example User Task Model

- Choosing and enrolling in courses for M.Sc. degree in CISC

Informal Task Model

- Figure out what courses I want to take
- Find out how many courses I should take per term
- Find out about professors
- Find out what term courses offered
- Find out what time courses offered
- Find out about Ph.D. breadth rules
- Discover conflicts
- Prioritize conflicting courses
- Enroll in courses
- Enroll in a course
- …

Heuristics

- Goal is to discover all tasks
- Tasks may be at different levels
  - Some tasks part of other tasks
- Express tasks from user's point of view
- HTA will organize tasks properly

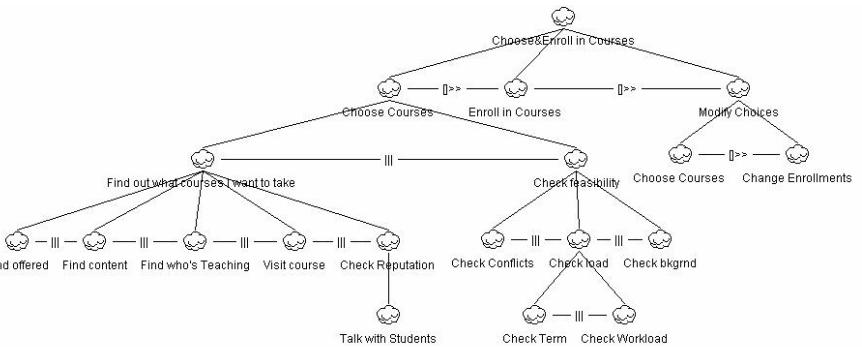
Informal Task Model

- Figure out what courses I want to take
- Find out how many courses I should take per term
- Find out about professors
- Find out what term courses offered
- Find out what time courses offered
- Find out about Ph.D. breadth rules
- Discover conflicts
- Prioritize conflicting courses
- Enroll in courses
- Enroll in a course
- …

Hierarchical Task Analysis

- Structure tasks into hierarchy representing task decomposition
- Identify relations between tasks
  - Order in which they must be performed
  - Information flow between tasks
- Identify optional ways of decomposing tasks

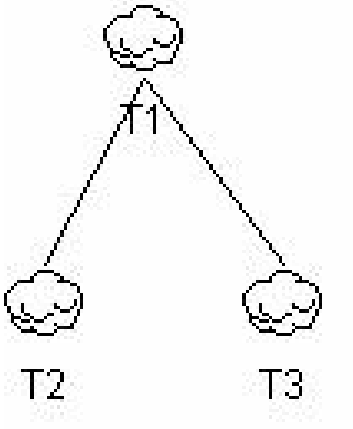
Hierarchical Task Model for Course Selection+Enrollment



Hierarchy

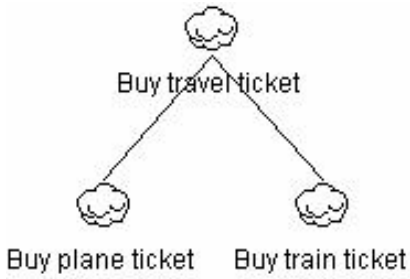
- Hierarchy represents how tasks are decomposed into subtasks
  - Read levels as "In order to do T1, I need to do T2", or "In order to do T1, I wish to do T2"

Hierarchy



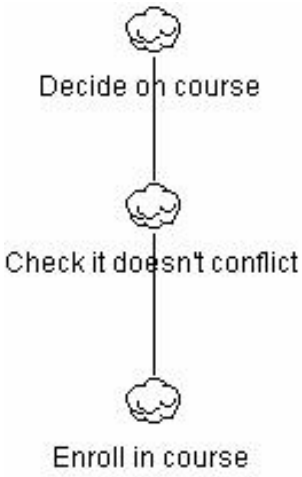
- Tasks at same level represent different options or different tasks that have to be performed
  - Read levels as "In order to do T1, I need to do T2 and T3", or "In order to do T1, I need to do T2 or T3"

Common Errors



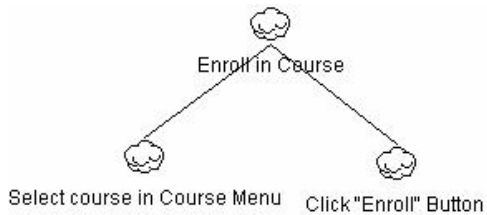
- Hierarchy does not represent abstraction
  - Hierarchy represents task decomposition
  - "In order to buy a travel ticket, I need to buy a plane ticket"

Common Errors



- Hierarchy does not represent sequence
  - "In order to check a course doesn't conflict, I have to enroll in the course"

Common Errors

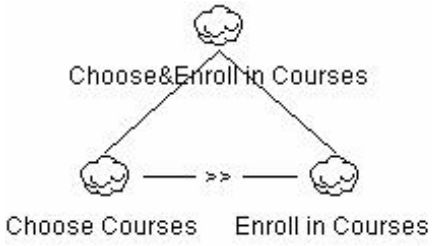


- In User Task Model, tasks do not represent specific interactions with system
  - Later we'll look at system task models where such tasks are modeled, but that follows UI design

Temporal Relations

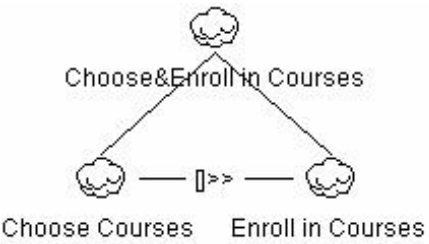
- Relations between tasks capture
  - Where tasks are required vs optional
  - Where tasks must be in a specific order vs can be performed in any order

Enabling



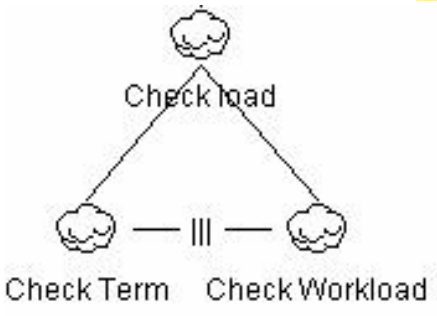
- Specifies second task cannot begin until first task performed
- I.e., I cannot enroll in my courses before I've chosen which courses to take

Enabling with Information Flow



- Specifies second task cannot be performed until first task is performed, and that information produced in first task is used in second

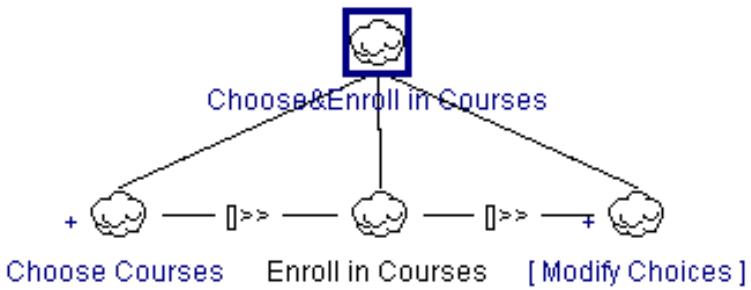
Interleaving



- Tasks can be performed in any order, or at same time
- In order to check the load of a set of courses, I need to consider what terms they fall in and to consider how much work each course represents
- I can do this in any order

Optional

- Some tasks need not be performed to achieve the goal task



Summary of Task Modeling

- Use activity analysis, ethnographic study, discussion with users to understand user tasks
- Brainstorm informal list of tasks
- Arrange and refine tasks, organizing into hierarchy
  - Write descriptions of tasks
- Define temporal relationships between tasks

In-Class Exercise

- Develop user-task model for ordering a pizza