User Centered Design
Mobile Interaction Design

21W.780 – Class 6
March 20, 2007
Frank Bentley

Today:

Goal of today's class:

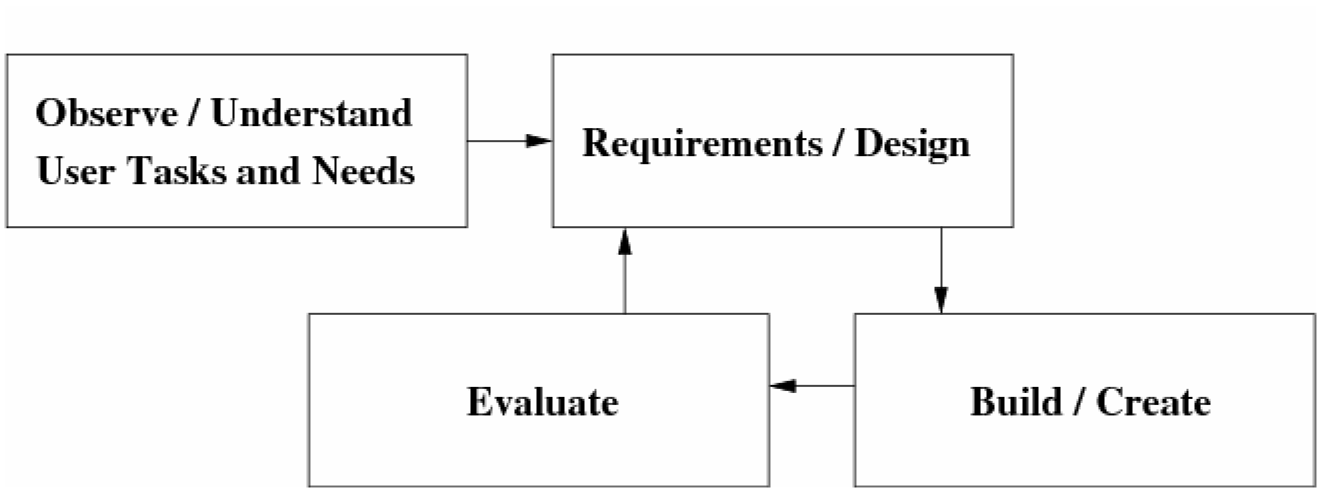
Think about your application as helping a user accomplish a set of tasks

This about each task comprised of actions along an interaction flow

Organize actions into screens, focus areas of interaction

Think about how to design screens, different mobile UI paradigms

User-Centered Design Loop:



Contextual Inquiry / Ethnographic-Style Observation

Contextual Design (Beyer and Holtzblatt, 1998)

- **A process of developing user requirements by understanding user behavior**
- **Involves observing users performing tasks similar to those they would be performing with your system**
  - People cannot be relied on to tell you what they think or how they approach tasks
  - In context, people can relate what they are currently thinking ("think aloud" methods, probing questions)

Contextual Inquiry / Ethnographic-Style Observation

Who to involve

- **Users most similar to those who will be using your system**
- **As diverse a set of users as you can get (age, gender background, lifestyle, tech usage, etc.)**
- **7-10 users is typically enough, stop when you keep seeing the same things**

What to observe

- **Tasks people perform / steps performed in those tasks**
- **Critical Incidents (things that don't go as expected or things that are exceptionally good)**

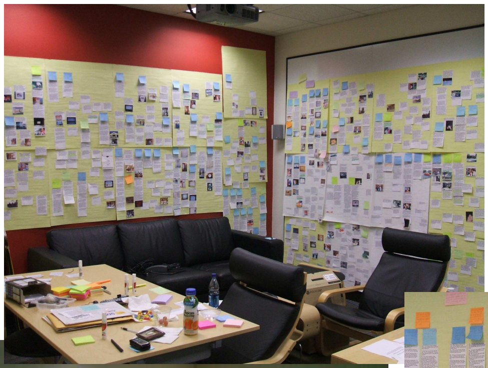
Affinity Diagrams…

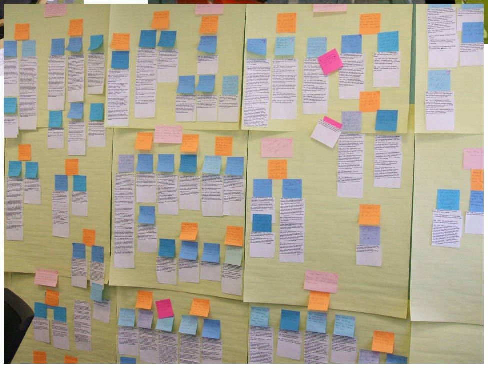
Way to visually organize qualitative data

Based on K-J and Grounded Theory analysis

Find themes and patterns in data

Hierarchical structure leading to holistic explanations





CI Models

Models summarize user behavior

Aggregated models across all participants can help in design

Artifact Model

- Capture information/things user interacts with

Cultural Model

- Capture people users interact with to complete tasks

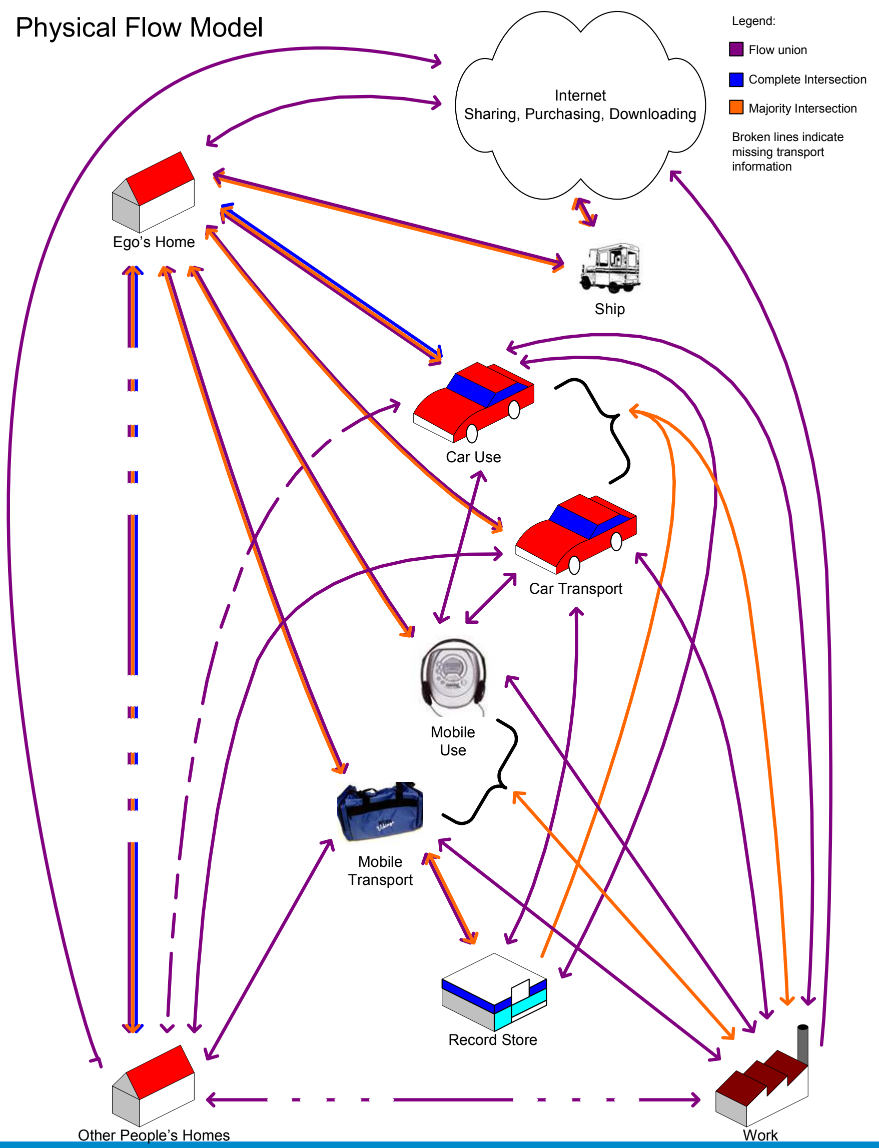
Physical Model

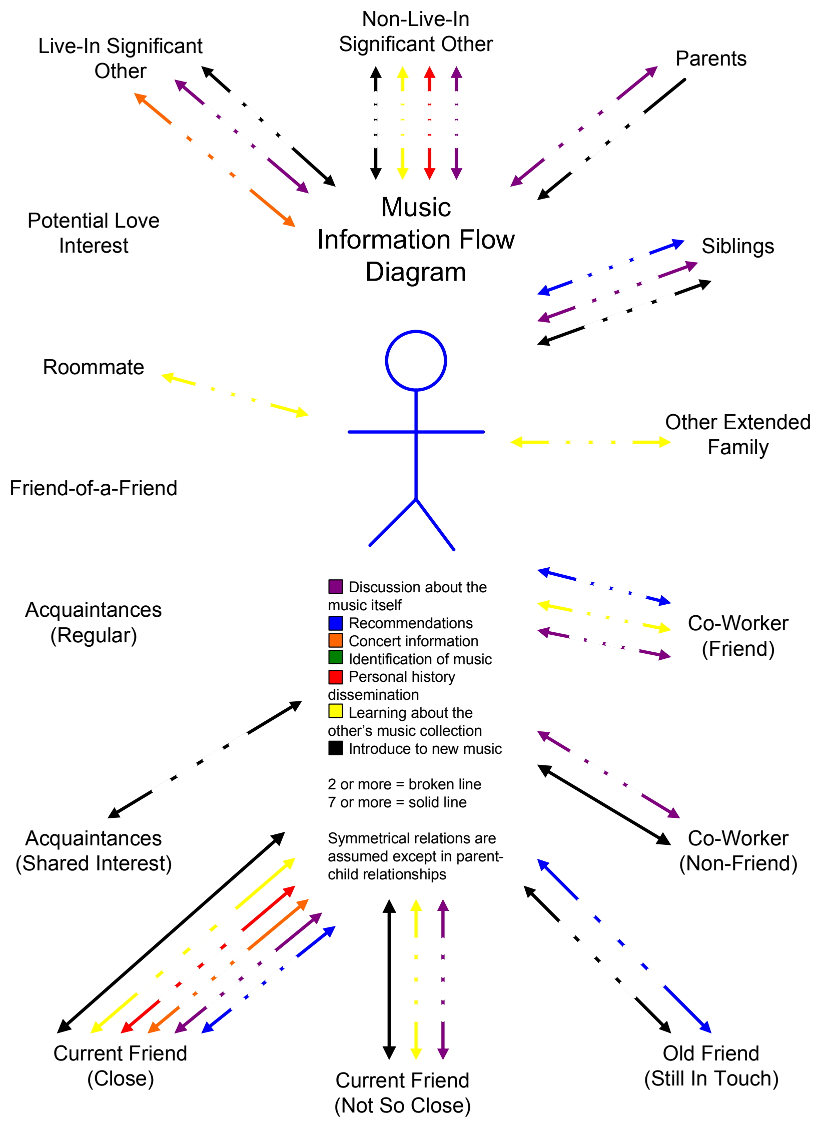
- Capture how people move and interact with a space

Sequence Model

- Capture tasks and steps performed to complete the tasks

Examples:


Physical Flow Model
Internet
Sharing, Purchasing, Downloading
Legend:
Flow union
Complete Intersection
Majority Intersection
Broken lines indicate missing transport information
Ego's Home
Ship
Car Use
Car Transport
Mobile Use
Mobile Transport
Record Store
Other People's Homes
Work


Live-In Significant Other
Non-Live-In Significant Other
Parents
Potential Love Interest
Music Information Flow Diagram
Siblings
Roommate
Other Extended Family
Friend-of-a-Friend
Acquaintances (Regular)
Discussion about the music itself
Recommendations
Concert information
Identification of music
Personal history dissemination
Learning about the other's music collection
Introduce to new music
2 or more = broken line
7 or more = solid line
Symmetrical relations are assumed except in parent-child relationships
Co-Worker (Friend)
Acquaintances (Shared Interest)
Co-Worker (Non-Friend)
Current Friend (Close)
Current Friend (Not So Close)
Old Friend (Still In Touch)

Task list (Use cases)

-Tasks are high level concepts of purposeful use

- Most revolve around end states the user would like to be in (e.g. “select music to play” “play desired music” “stop playing music” “add music to collection”)

- Not individual requirements for a system

- Ideally, tasks come from observations, user needs

Task list (Use cases)

Make a task list for you application

Requirements list

Functional requirements needed to accomplish tasks

Can be user facing (visible) or system facing (hidden)

Should exhaustively enumerate everything the application/system has to perform

Prioritize list to determine what will be implemented / what can safely be omitted in early versions (common prioritizing is Core, Important, Nice to Have)

Prioritize by use (Used by many, most, few) and expected frequency (Used often, sometimes, rarely/once)

You’ll rarely be able to implement everything or cleanly fit it into a design

Requirements list

Develop requirements for each task…

Select music to play:

Select metadata attributes to search on

- **Search on Artist**
- **Search on Album**
- **Search on Genre**
- **Search on Playlist**
- **Search on Year**

Search on combination of attribute/values

View values for the given attribute

Select an attribute

View songs matching the query

Play entire results list

Play starting at an item in the results list

Requirements list:

Make a requirements list for you application

User Environment Diagrams

Represent groups of tasks / requirements that the user will perform into “focus areas”

Shows links between areas

Begins to approximate user interface

Each area is meant to represent functions and objects of interaction required for a particular type of work

For each area list:

- **Purpose (summary of why the user would be in this state)**
- **Functions (list of available functionality in this state)**
- **Links (list of places the user can navigate to from here)**
- **Objects (things the user can see and interact with here)**

Hidden areas can represent tasks done by the system

User Environment Diagrams

Playing Music

Purpose: This area allows a user to control the playback of music in a playlist.

Functions:
Pause Music
Skip Track
Go back a Track
Skip to a given track in the current playlist
Adjust volume

Links:
Change Playlist
Create new Playlist
Exit Application
Hide Application

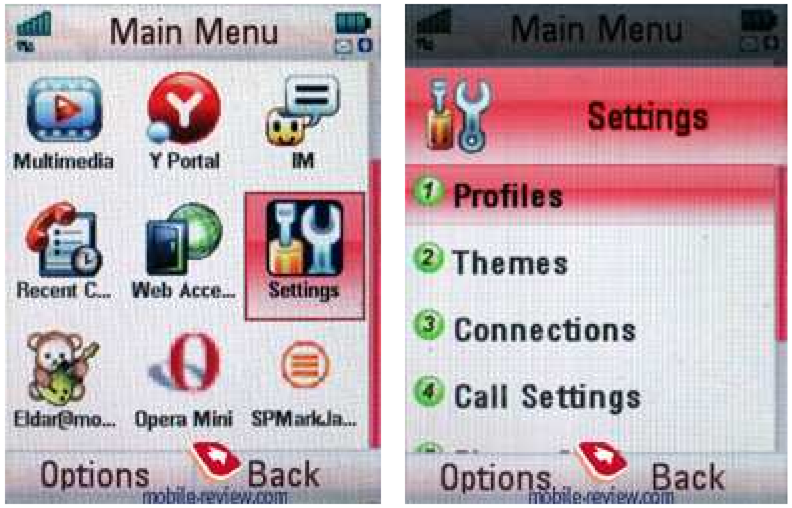
Objects:
Current Playlist

User Environment Diagrams

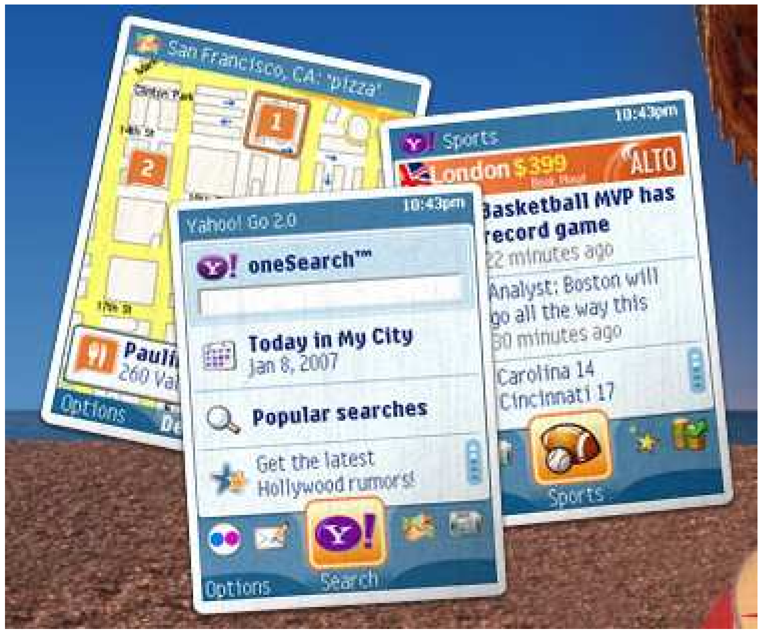
Create a User Environment Diagram for your applications

Mobile UI Paradigms

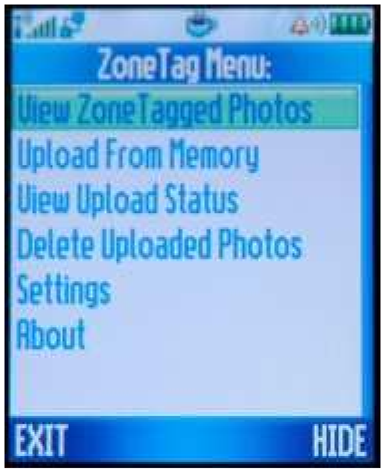
Clustered List:



Carousel:



Task-Based:



Mobile UI design considerations…

Click count is important, but good default options are more important

Minimize need to scroll – multiple screens often better than one long scrolling screen

Consistency – “Back” and Confirm actions always in the same place (J2ME takes care of this if your Command objects use the proper type e.g. Command.BACK, Command.OK)

Shortcuts for advanced users – e.g. GMail’s number shortcuts for delete, compose, etc.

Heuristic Evaluation

A list of common errors in user interfaces to check your interface against (sanity check)

Simple way to evaluate interface without involving formal user study

Can generally solve many initial usability issues

No replacement for usability testing

Nielsen's Heuristics

Visibility of system status

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

Match between system and the real world

The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

User control and freedom

Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

Consistency and standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

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.

Recognition rather than recall

Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

Flexibility and efficiency of use

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.

Aesthetic and minimalist design

Dialogues 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.

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.

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.

Usability Testing

Best way to learn how interface will be used is to see it used

Choose tasks that users would actually perform (don’t ask someone to do something they never intend to do)

Use 5-7 users to catch majority of major flaws

Tell user that interface is being tested, not them

Have users “think aloud” verbalizing what is going through their heads, not reflections on what they are doing

Don’t help users (only ask them to keep talking or move to the next task upon success / failure)

Determine ahead what constitutes a failure case, don’t allow users to run amok in your UI aimlessly

Watch for critical incidents

References

Contextual Inquiry / CI Models / User Environment Diagrams

Beyer, H. and Holtzblatt, K. 1999. Contextual design. *interactions* 6, 1 (Jan. 1999), 32-42. DOI= http://doi.acm.org/10.1145/291224.291229

Beyer, H. and Holtzblatt, K. 1998 *Contextual Design: Defining Customer-Centered Systems*. Morgan Kaufmann Publishers Inc.

Tasks and Requirements Analysis

Ellen Isaacs , Alan Walendow , Alan Walendowski, Designing from Both Sides of the Screen: How Designers and Engineers Can Collaborate to Build Cooperative Technology, New Riders Publishing, Thousand Oaks, CA, 2001

Paper Prototyping

Rettig, M. 1994. Prototyping for tiny fingers. *Commun. ACM* 37, 4 (Apr. 1994), 21-27. DOI= http://doi.acm.org/10.1145/175276.175288

Heuristic Evaluation

Nielsen, J. 1994. Heuristic evaluation. In *Usability inspection Methods*, J. Nielsen and R. L. Mack, Eds. John Wiley & Sons, New York, NY, 25-62.

Next Steps…

Discuss last week's assignment – image capture

Progress update on projects

Due 4/3:

1) Create a paper prototype of your application