TOPICS:

• Ethnographic field studies
• High concept statement
• Needs analysis
• User analysis
• Task analysis
• Usability goals
INTRODUCTION TO SYSTEMS ANALYSIS

• Revisiting the usability engineering life cycle
ETHNOGRAPHIC FIELD VISIT

• In anthropology and sociology, ethnography is:
  * Participating, “overtly or covertly, in people’s daily lives for an extended period of time, watching what happens, listening to what is said, asking questions” [Hammersley & Atkinson 1983, as quoted in Shneiderman, p. 107]

• For user interaction requirements gathering:
  * UI designers limit study to days or even hours, but have to obtain needed data

- "Quick and dirty ethnography"

• Goal of ethnography for UI designer is to collect information needed to ensure usability of design

* The "User Interface Requirements Detective"
ETHNOGRAPHIC REQUIREMENTS
GATHERING

• Process for UI ethnography includes:

* Preparation for field study
- Start with own “brainstorming” of user task statements
- Understand organization’s policies and culture
- Check out their website
- Know current systems and history
- Prepare script of questions for interview
- Select appropriate users to observe and/or interview
ETHNOGRAPHIC REQUIREMENTS
GATHERING

* Perform field study
- Establish rapport with managers and clients
- Observe and/or interview users in workplace
- Collect quantitative and qualitative data
- Collect artifacts (e.g., paper forms) as available
- Follow leads from visits, if any
- Document and characterize user classes
- Document work flow/user task analysis

* Keep focus of activities user-centered!
ETNOGRAPHIC REQUIREMENTS
GATHERING

• Seems easy, but it’s not!
  * Hidden traps, surprises (e.g., what to wear to interview, different perceptions of managers vs. users, different use of language/technical terms)

• Equally important as data collected: rapport and relationships with client, users established during process

• What if client is reluctant to give access to users?
  * Ask for just a couple of hours
  * Establish *necessity* for usability
INTRODUCTION TO EXAMPLE SYSTEM

• Y2K Calendar System

* Simple automated version of a paper calendar

* Goal is to learn the development process, not to produce a marketable calendar product
HIGH CONCEPT STATEMENT FOR PRODUCT

• High concept statement — brief descriptive summary of product, typically 50 – 75 words

• Mission statement for a product, to help focus product development

• Writing a good high concept statement is not easy and is not done once; highly iterative

• Answer following questions:
  * What is product name?
  * Who are product users?
  * What will product do?
  * What problem will product solve?

Automated management of appointments
HIGH CONCEPT STATEMENT FOR PRODUCT

• A possible high concept statement for Y2K Calendar:

* Our calendar will have automated support for scheduling appointments, to improve customer satisfaction.

Too vague

• A better high concept statement (47 words):

* The Y2K Calendar will allow a broad variety of users to schedule and manage appointments. These users can range from professionals using the system to run an office to casual users keeping track of personal information. Automated support will reduce scheduling effort and increase awareness of appointments.

E.g., reduce scheduling by supporting recurring appointments.

E.g., increase awareness by giving alarm (visual and/or audible)
EXAMPLE: SYSTEMS ANALYSIS

• **Goal:**
  * To make a fast tour through the process of determining basic user and system requirements

• **Activities:**
  * A sampling of needs analysis, user analysis, task analysis, usability goals, and requirements definition

Dilbert: engineers ->->->

3.10 SA
EXAMPLE: NEEDS ANALYSIS

• *Goal of system:* Manage appointments

• *Assumption:* Some boundaries set by management, marketing, customer, etc. (e.g., hardware); determination made that product is novel, market not yet saturated

• *Features*

  * Appointment means information on:

  Date
  Time
  Place
EXAMPLE: NEEDS ANALYSIS

Appointment description

* Manage means:

Add new appointment

Delete existing appointment

Modify existing appointment

* Plus, need ability to view/display appointments

Task = user; function = system

• Follows from user analysis, not just developers’ ideas
EXAMPLE: NEEDS ANALYSIS

- After observing users, someone thinks of “alarm” idea

  The needs don’t come all at once, up-front

  * Do we want to actively inform of appointments?

    Maybe ask or observe users.

  * Decision: Yes, very useful

  * Iterate and revise needs

- New feature: Active reminder

  Increased functionality
USER ANALYSIS

• User classes are about roles, not individuals

• “Know thy user” — and it is not you!

* Important to have representative user(s) on development team and/or have access to representative user(s)
USER EXPERTISE LEVEL

• Expertise levels do not define user classes, but occur within user classes

* Novice or first-time user: May know application domain but does not know specifics of application

* Intermittent user: Uses several systems from time to time; knows application domains but does not remember details of different applications

* Frequent user: “Power” user who probably uses application daily and knows both application and task domain very well

• Design may have to account for each of these expertise levels

These are not the specific user class types you should identify for your project!
EXAMPLE: USER CLASS ANALYSIS

• What are characteristics of users of Y2K Calendar system?

* General characteristics

- Busy people
- Keep schedule for self and others
- Professional and personal use
- Calendar is very small part of job
- Need ‘transparent’ tool
- High general skill level, literate
EXAMPLE: USER CLASS ANALYSIS

* Domain skills
  - Know how to use calendar

* Computer skills
  - Broad range
  - At least some typing skills
  - Familiar with GUI/mouse
EXAMPLE: USER CLASS ANALYSIS

• Conclusions

* Keep it simple
* Usability as important as functionality (or more)
* Try to get functionality greater than paper calendar
* Try to get usability greater than paper calendar
* Minimize typing
* Users must learn it quickly

• Caution: Difficult for users to tell developers what they want or need
* Important to observe users in their typical work environment

Developer: I try to tell users what they need, but they don't want to listen to me
## EXAMPLE: USER CLASS CHARACTERIZATION MATRIX

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>User Class A</th>
<th>User Class B</th>
<th>User Class C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Education/skill level</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computer knowledge/Experience</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application domain knowledge</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage frequency, duty cycle</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complexity of domain content</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User interaction with others</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discretionary or captive</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Training needs/opportunities</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Culture</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receptive/resistant</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TASK ANALYSIS

Dilbert: psychic ->->->

• Hierarchical task analysis (HTA)
  * Structure, organization, and relationships of tasks users perform with system
  * Not timing, precedence, order of task performance, work flow, etc.
  * Only what user can do, not must do

• What developers observe that users need, not what developers think that users need

---Larsen: used car salesman, Do not press button

• Task analysis is probably the most overlooked and shortchanged activity in the whole user interaction development process
TASK ANALYSIS

• Hierarchical task decomposition

* Task names: <action object>

* Lower levels collectively represent their "parent" in the tree

* Hierarchical task analysis does not show sequencing
EXAMPLE: TASK ANALYSIS

- What tasks will users perform with this system?
  
  * Tasks are performed by user (e.g., view)
  
  * Initial list of major sub-tasks
    
    - Add new appointment
    
    - View existing appointments
    
    - Modify existing appointments
    
    - Delete existing appointments
    
    - Set alarm
    
    - View calendar
    
    - User-centered wording, not system-centered (e.g., view appointment, not display appointment)
EXAMPLE: TASK ANALYSIS

- Task analysis iterated

  * As think about viewing appointments, realize the need for different levels or scopes of view

    - For example, by month, by week, by day, by hour

    - Implication: Add “control view” task to list

  * Also discovered need to search appointment database to retrieve by content

    - Implication: Add to needs, tasks, functions, requirements

    - Note: From here on, "requirements" means interaction design requirements

But can't separate entirely from system, functional requirements
EXAMPLE: TASK ANALYSIS

- Task analysis iterated (continued)

* Another example of iteration: Alarm feature will lead to user tasks (to set parameters)

- Decision: For now, hard wire for 10 minutes before appointment; no user tasks

Good example of early simplistic design decision; needs iteration
EXAMPLE: TASK ANALYSIS

Example of possible quasi-hierarchical user task structure for Y2K Calendar

- Structure diagram is accompanied by brief description of what each box means
USABILITY GOALS

- High-level objectives in terms of usability and design of user interaction
  
  * Reflect real use of product in real world
  
  * Determine what is important to organization and to users

  Learnability for new users; power performance for expert users; avoiding errors

  * Usability goals can be driven by marketing

- Usability evaluation design driven by usability goals
USABILITY GOALS

• Determining usability goals in terms of
  * User classes
  * User task context, special tasks
  * Walk-up-and-use learnability
  * High performance for expert users
  * User errors
  * User satisfaction
USABILITY GOALS

• Example usability goals for Y2K Calendar:
  * Fast walk-up-and-use for simple tasks
  * High learnability for more advanced tasks
  * Low error rates for rescheduling appointments
  * Increase effectiveness of calendar by helping users avoid missed appointments

Systems analysis: tree -&gt;&gt;&gt;&gt;