Usability and GUI Design and Principles
Part I: Principles, Usability, Design, User Centered Design Process

Shihong Huang
shihong@fau.edu
Florida Atlantic University
Acknowledgment

- This lecture is based on the lecture note of Professor Dragutin Petkovic at San Francisco State University (SFSU), “Usability, GUI Design and Principles”
Objectives

• Learn how to design and test easy to use systems: process (User Centered Design, focus groups, usability testing, iterations) and principles (GUI design principles, design patterns)

• NOT on how to program GUI but on how to design it from scratch
GUI – Graphical User Interface

- GUI is part of Human-Computer Interaction (HCI) which is the study, planning and design of how people and computers work together.
- User Interface (UI) is what users see, hear, touch, talk to or control and direct.
- UI is often “the product”. This is what users see and may make their decision on!
UI and GUI are everywhere

- PC
- Browsers
- Mobile
- TVs
- Phones
- Airplanes
- Cars
- Appliances

GUI is critical component of most systems
Usability – more than GUI

- Usability refers to ease of use of the product, and includes GUI but also everything else that “touches” the user:
  - Documentation
  - Installation
  - Purchase, registration
  - Support
  - API and SW tools usability
- For non-SW products:
  - Packaging
  - Assembly
  - Ergonomics (physical)
  - ....
- There are formal methods to assess usability
Importance

- Increasingly critical for product success (e.g. Apple/Samsung patent battle)
- Users expect systems to work but will choose those that are easier to use
- Industry looking for SW engineers who can provide great “user experience”
What is good GUI, what is bad GUI?

- Class? Good and bad
- To answer this ask yourself first:
  - Who is the user
  - What are users’ tasks and goals
Your reaction and actions to bad GUI

- If part of your work-related apps:
  - Stress, fatigue, panic, frustration, annoyance
  - Low productivity, errors → increased cost of doing business

- If for personal use:
  - Frustration, annoyance
  - Return, never use it again
  - Abandon WWW site or transaction
  - Never buy product from that company
  - .....
Your goal: Total User Satisfaction

Total user experience: from “opening the box” to calling support

- Features
- User Interface
- Response Time
- Reliability
- Installability/Deinstallability
- Help and docs
- Maintainability
- Graphics/attractiveness
- ....
Your goal

- Make systems that have all three:
  - Useful
  - Usable
  - Used

Examples:
Benefits of good GUI – examples from internal business apps

- Say one application involves users going through 4.8 million pages/year. Just 1 second savings per page saves 0.7 person/year of work
- Screen users in one study were 20% more productive with better screen design or performed transactions in 25% less time with 25% less errors etc.
- One study of WWW shopping (Fath 1999) showed that for better designed sites task completion was 65% higher ➔ $$$$  
- Class experience?
Benefits of good GUI – example for WWW

- You look for an item via WWW on-line shopping
  - Site A: confusing, you are lost….why use it, go somewhere else (otherwise good selection and price)
  - Site B: easy to use, good on-line experience ➔ they got the business
Benefits of good GUI – example for mobile

- You are buying a mobile gadget
  - Product A: has good price, ton of features BUT does not look cool, hard to activate, packaging is boring, feels cheap and shabby, has confusing UI
  - Product B: cool, including packaging, delivery, easy to activate, feels good in your hand, easy to use by touch and hints and gestures ➔ winner

NOTE: traffic from mobile devices will surpass traffic from PCs in 2012
GUI, usability for the WWW

- Users can go elsewhere, they have many alternatives ➔ Poor site means lost business (vs. users who bought MS Word – they usually stick with it)
- Users do not want to be trained or read user guide ➔ has to be easy to use and intuitive

This is different from applications you a) buy or b) are forced to use at work (e.g. SFSU HR apps)
GUI: Users View

GUI “toolbox”
- Windows
- Folders
- Tools
- Menus
- File
- Paste
- Drag & Drop
- ......

Tasks
- Create a report
- Fill in the database
- Order goods
- Make a reservation
- Find the movie

Users
- Professional at work
- Home use
- Skills/education
Some fundamental observations

- UI and usability involves people and is for people ➔ GUI design principles, design patterns leading to implementation libraries/widgets
- Careful design with user in the loop has to precede coding
- Design/Testing/evaluation has to involve human subjects other than designers
- UI code is large, not algorithmically deep but very detailed. Its development/QA is often grossly underestimated and consists of many small simple steps
- Text/language on the screen is critical
UI from different perspectives

**User view:**
- Satisfies user needs and his/her capabilities in the best possible way
- Not noticed, easy to learn, intuitive
- Permits user to focus on tasks and information not the mechanism of the GUI
- Fast response
- Polished, attention to every detail
- Good text – focused, task oriented, not too long, not too short
- Works same on all browsers, colors, resolutions
- Attractive graphically
- What else?

**Developer and management view**
- Built on time, within the budget
- Sells well
- Users and press happy
- Easy to maintain
- Works/adapts to variety of delivery devices (various browsers, different screen sizes, PCs and handheld)
- Easy to support
- What else (class)?
GUI in products

- Poorly designed GUI is not useful, no matter how well the product performs
- Well designed GUI is also not satisfactory if it is not reliable

⇒

- Have limited set of functions that is designed and implemented well rather than a lot of poorly designed and unreliable ones
User-Centered Design and Development – **MUST DO for GUI**

- **Conceptual design, prototype, Specs, style guide**
- **High-level design, Prototype, specs**
- **Low-level design, Prototype, specs**
- **Code and unit test**
- **System test**

**Requirements**

**Plan**

**EVALUATE**
(users, management, Support, sales, QA...)

**Iterative, evaluated all the time by multidisciplinary team**
How to guide GUI design and development

- Many way to design a good GUI (and many ways to design a poor GUI) – no single solution that is optimal
- Use GUI design principles and patterns for overall design – they are proven solutions. Some specific guidance is provided by some manufacturers (e.g. Apple, Microsoft)
- For implementation use available or required widget libraries as much as possible (tested for usability, portability and implementation: e.g. calendar, search menu ...)
- Use UCD for overall SE process ➔ test often and at every stage using typical users (not designers of GUI)
Check excellent Apple guidance – follows our class very well

- iOS Human Interface Guidelines describes the guidelines and principles that help you design a superlative user interface and user experience for your iOS app
iPhone Lineup

www.apple.com
One design pattern: Structure of WWW pages

(This is also one way of high level UI spec)

<table>
<thead>
<tr>
<th>Standard browser buttons (constant)</th>
<th>Advertisement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Tabs for context</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>Related links</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Browsable File List for given context</th>
<th>Main area (info, entry fields, links, Search results...)</th>
</tr>
</thead>
</table>
Mobile

- Mobile traffic to surpass PC traffic in 2012
- Hot platform of choice for programming
- Characterized by: small display, different input/output, smaller memory, CPU, smaller bandwidth, often interrupted
- Usability is critical: these devices are very "personal" and are used in a specific way
- Has its own specific:
  - GUI design principles
  - GUI design patterns
  - GUI programming environment, tools etc.
Mobile devices


- Discuss what is unique/different here vs. PC and WWW for GUI/app design
GUI Design and Implementation

- No fixed and strict rules – there are many ways to design good (or bad) UI
- BUT, there are GUI design principles which are derived in great part based on innate characteristics of humans sense and perception
- **GUI Design principles** are to be used as a guidance and “test”
- **GUI Design patterns** embody experience of successful UI designs – serve as a more specific guidance for the design of GUI elements and systems
- **GUI widgets** – prefabricated components used to build GUI, tested for usability and cross-browser support (e.g. calendars, menus etc)
From principles to design patterns

- GUI design principles are embodied in GUI design patterns: cookbook advices on how to solve particular GUI issues (context, tool tips etc.). In same cases provide code (e.g. GUI widgets)

- Examples for WWW:
  - Yahoo Design Pattern Library:
    http://developer.yahoo.com/ypatterns/
  - Build User Interface:

- Examples for mobile platforms:
High level Principles of User Interface Design

- General principles to be followed for all UI designs.
- Derived from inherent cognitive/perceptual human properties
- List given in alphabetical not priority order
- All principles are important, BUT relative importance might depend on users and business objectives

Have this as a check list to design/evaluate GUIs
Aesthetically pleasing

- Contrast between elements (see our WWW links in PP)
- Create groupings
- Align screen elements and groupings
- Provide 3-D feel if possible
- Use colors and graphics appropriately
- (Make sure your choice of colors and graphics renders in all browsers, all screen and monitor color resolutions etc.)

This is not l’art pour l’art – it does make difference.
This will be covered in GUI design guidelines
Often specified in “style guide”
Clarity – visual, conceptual, language

- Visual elements
- Functions
- Metaphors
- Words and text

- Error message and text in general are often very confusing
- Example: Form filling, purchasing: E.g. Am I entering my name or someone else’s name? Whose address goes here?
Compatibility

- With the user characteristics
- With User’s mental concept of the application
- With task and job to be done
- With the product
- With rendering device (PC vs. handheld)

*Do not assume all users are alike or that all are like developers*

*User-centered design helps*
Configurability

- Allow personalization and configuration of settings
  - Gives users sense of control
Consistency

- System should act, look and operate in consistent ways.
- Similar components should have similar look, similar uses and operate similarly
- Same actions should always produce the same result
- Function of elements should stay constant
- Position of standard elements should not change
- Use same font for similar level of abstraction

Consistency of Window and MAC UI helps a lot
E.G. Right Mouse Clicks and Left Mouse Clicks on Windows, behavior of Windows containers etc.
Control

- User controls the interaction
- Do not let the system think it knows what is best for the user
- Do not perform “silent” actions
- Actions should be capable of interruption
- Maintain user-centered context
- Avoid modes (impose constraints, hard to keep in context, and complicates QA)
- Allow customization, always have defaults
Context

• User should know the context all the time. This is especially challenging for Web UIs
• User should never get lost
• Should always be able to easily go back and go home
• For search: always know what you search for, where you applied the search for
• For applications consisting of several ordered steps (shopping etc.): same as above, plus know what is the next step, easy to abort at any time
• In navigation and browsing: show the lineage (see new Apple browser)
• In Windows one way of keeping context is multiple windows.
• Context is harder to do in the WWW. What means of keeping context in WWW are you aware of?

  Eg: VMware, Yahoo
Easy to understand

- What to look at
- What to do
- Why
- When
- How

Flow of actions should be easy to follow and learn and has to make sense
Efficiency

- Minimize eye and hand movements via proper screen layout etc.
- Fitts’ law: Time-to-acquire-target = a + b*Log₂(2A/W) (A is distance, W is width of the target)
- Anticipate user needs (but don’t act on them)
- Minimize number of clicks and paths for each task

Can result in huge savings, less user fatigue, increased productivity etc.
Familiarity

- Employ familiar concepts and language
- Keep the UI natural as much as possible
- Use real world metaphors
- Follow adopted and de-facto standards (Windows, Mac, WWW)
- Follow common styles for each technical area (Database admin, systems management etc.)

“Good copy better than poor original”
Functionality (not necessarily a graphical design issue, but I added it here)

- Does the UI have all the right functions for major tasks
- Are functions clear and non-overlapping
- Major functions and advanced ones
- Effective functions for novices v.s. power user
- New functions can be automated versions of common user tasks that could be done by other means

- Examples: VMware RR; Apple Snap-back for new browsers; Save and Add Next in Yahoo address book
Forgiveness, recovery

- Tolerate errors that are common (i.e. do not allow crashes)
- Prevent errors whenever possible
- Protect against catastrophic errors
- When error occurs, provide constructive message
- Commands should be able to be abolished or reversed
- Work should never be lost in case of user or system error
- Warn the user in case the operation causes some dramatic change or possible error (sometimes make it a hint, sometimes not)
Predictability

- User should be able to anticipate natural progression in the task (via screen elements, cues)
- Important for applications where series of steps needs to be done (workflow, WWW shopping)
- User should know all the time where he/she is, what is the next step, how to go back, how to abort etc.
- Context is important
Responsiveness

- Systems must respond rapidly. Anything above 0.5 sec. is not considered interactive and causes user frustration after long usage.
- Response time should be consistent, logical and not vary considerably for similar functions.
- Classic client systems much faster than WWW. But, in WWW users seem to expect delays.
- Tricks can be used to avoid user problems for less responsive systems:
  - Show user something, anything but blank screen.
  - Show some visual clues that the system is working (e.g. hour glass cursor, progress bar, progressively rendered images...)
  - Avoid periods of blank screens.
- Generally, provide acknowledgement of user actions:
  - Visual
  - Textual
  - Auditory

This is directly related to user satisfaction, usability and productivity.
Simplicity

- Create as simple GUI as possible (e.g. Google)
- Use progressive disclosure
  - Show common and necessary functions first
  - Prominently feature most important functions
  - Hide advanced functions
- Provide defaults
- Reduce screen complexity
- Make common actions simple to use, even at the expense of less common actions being harder to use

- Examples: www.google.com
Transparency

- Permit the user to focus on the task
- Mechanics of interface should not get in the way
- Workings and reminders of workings inside the computer should not be visible to the user
- No silent action (e.g. doing something important without users being informed...)

Florida Atlantic University  Shihong Huang
Design trade-offs

- Final GUI will be based on trade-offs, often balancing conflicting design points
- User requirements always come above technical, development or engineering requirements
How do you rank importance of GUI design principles?

- Aesthetically pleasing
- Clarity
- Compatibility
- Easy to understand
- Configurability
- Consistency
- Control
- Context

- Efficiency
- Familiarity
- Functionality
- Forgiveness, recovery
- Predictability
- Responsiveness
- Simplicity
- Transparency
- Design trade-offs
Dragutin’s top

- Simplicity
- Functionality
- Context
- Consistency
- Responsiveness
- Layout
Examples (good and bad)
UI – what not to do
Confusion between *state* and *action*

**Power Off**

Is this the current **STATE** of the system OR it **asks** the user to click to **POWER-OFF** (e.g. system is currently in **ON** state)

**System ON**

**Power Off**

Solution: make the state clear
Asks semester entry before login, login required for each new semester entry.
What you see is NOT what you get
After clicking Here the screen Looks the same

The results are below, Needs SCROLL to see
USPTO Patent Full-Text and Image Database

Data current through August 2, 2011.

Query [Help]
Term 1: [ ] in Field 1: [All Fields]
AND
Term 2: [ ] in Field 2: [All Fields]

Select years [Help]
1876 to present [full-text]

[Search] [Reset]

Patents from 1790 through 1975 are searchable only by Issue Date, Patent Number, and Current US Classification. When searching for specific numbers in the Patent Number field, patent numbers must be seven characters in length, excluding commas, which are optional.
Some good designs
Alignment, layout, consistent use of font, help/explanations obvious.

The image shows a screenshot of the Advanced Patent Search interface on Google Patents. The interface includes several fields for searching patents, such as Patent number, Title, Inventor, Assignee, U.S. Classification, International Classification, Patent type/status, and Date. There are options to search with all of the words, with the exact phrase, with at least one of the words, or without the words. Each field has a specific purpose, and the interface allows users to refine their search by specifying patent numbers, titles, inventors, assignees, and other criteria.

Florida Atlantic University
Shihong Huang
Alignment, layout, consistent use of font, help/explanations obvious

Nice grouping
Design pattern for showing context in browsers – (where am I, what I did, what I need to do next) – use STEP1, 2… useful when user has to do some steps in predefined order. (From Yahoo)

Also nice grouping into logical units
Clearly says WHAT the apps do, let you TRY it, then says how to START
Usability and GUI Design and Principles
Part II: Use cases, about the users, usability examples, case study
User Characteristics

Know your user, client, customer!!!!

• First, some general observations about the user
  • How people interact with computers
  • Human characteristics of relevance to GUI

• Usability testing (not only for GUI)
Important Human Characteristics related to GUI: *Perception*

Awareness and understanding of elements and objects through sight, sound.... Influenced in part by experience

- *Proximity*: objects relate to each other
- *Similarity*: similar objects should relate
- *Simple* designs are easier to remember
- *Closure, unity*: objects forming closed shape perceived as groups
- *Balance and symmetry*: vertical and horizontal
- Subjective perception of motion, image, video, audio quality
- *Context*: color backgrounds influence color perception etc.

➤ Implications for GUI screen and element design (to be covered later)
➤ Drives GUI design patterns
Examples of common graphic signs
Examples of visual illusions

- From: [http://dragon.uml.edu/psych/illusion.html](http://dragon.uml.edu/psych/illusion.html)
  - Experience Your Blind Spot
  - Figure of a Woman
  - Hering Illusion
  - Howard's Color Contrast
  - Horizontal-Vertical Illusion
Important Human Characteristics related to GUI: *Memory* (Miller, 1995)

- **Sensory memory**: buffer that stores sensory input. Forgotten easily.
- **Short-term memory**: holds limited amount of data for several minutes. Experiments showed that it holds *up to seven* concepts.
- **Long term memory**: large storage capacity, but can be unreliable.
Important Human Characteristics related to GUI: *Memory*

- **Recall**: getting information from long term memory. It is hard
- **Recognition**: selecting from multiple choice or recognizing a pattern. Easier
  
  ➔
  
  - Leverage speed of short-term memory by proper graphics and layout
  - Use pull down menus to “remind” people and help access long term memory
Important Human Characteristics related to GUI: Vision

- **Visual acuity**: capacity to resolve details
- Foveal and peripheral vision
- Relative acuity is halved at 2.5 degrees from the center ➔ influences character spacing on the screen etc.
- Eyes jitter (this increasing accuracy). However:
- Some patterns on the screen can seem jittery ➔ Care has to be taken in designing patterns so as not to cause jitter and visual distraction
- Busy periphery can distract main foveal vision ➔ implications on screen design
- Acuity drops with age ➔ Font type contrast and size to be adjusted for older users. Same for URLs
Important Human Characteristics related to GUI: *Mental models*

- Users have mental models of everyday things as well as established computing models (file systems, document model, transaction model, cars, TVs etc.).
- If GUI matches user or established mental model with the applications it is easier to learn, feels more intuitive etc.

→ **It is highly recommended to develop GUIs and applications with established models for the intended application and stick to them**
  - File/document model (new, edit, file, copy, modify)
  - Airplane simulator
  - Transaction model
Important Human Characteristics related to GUI: Movement Control - Fitts’s Law

- Keyboard, mouse, trackball usage involves movement
- **Fitts’s Law** (1954) is one of the rare exact laws related to usability:

  Time-to-acquire-target = \( a + b \times \log_2(2A/W) \)
  
  (A is distance, W is width of the target
  a and b are empirically determined constants)
**Movement Control**

- Large objects for important functions
- Group related buttons
- Pinning action of the sides, top, bottom and corners of screens
- Avoid small clickable objects
## Important Human Characteristics related to GUI: *Interactivity*

<table>
<thead>
<tr>
<th>Response Time</th>
<th>User reaction and perception</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 0.1 sec.</td>
<td>Instantaneous, no interruptions</td>
</tr>
<tr>
<td>&lt; 1.0 sec.</td>
<td>Delay noticed, but no break in thought stream</td>
</tr>
<tr>
<td>&gt; 10. sec</td>
<td>Interruption, user switches to another task</td>
</tr>
</tbody>
</table>
Important Human Characteristics related to GUI: *People differences*

- We are all different: motor and perception abilities, skills, education, learning abilities, age etc.
- How to design for wide population?
  - Lowest common denominator?
  - Provide varying levels of GUI that can be adjusted to various people
  - Allow people to customize
User/task considerations (for categorizing and surveying users)

Knowledge and Experience
- Computer Literacy
- System experience
- Application experience
- Task experience
- Education
- Reading level
- Typing skill
- Native language or Culture

Important
Types of users vs. skill level

Novice users

- Depend on systems features to assist (help etc.)
- Restricted vocabularies
- Simple tasks
- Need learning time
- Sometimes high “startup” curve that may lead to dropping the SW

➔ *Novice users are never “stupid”. Systems must be designed for them. They can be your key market segment especially in early stages.*
Types of users vs. skill level

Expert users
- Leverage experience
- Use recall (fast) rather than recognition
- Fast
- Need less help and feedback
- Like to reduce keystrokes, have shortcuts etc

→
- Allow experts to leverage their expertise. Systems also need to be designed to accommodate them
- How to design good UI for novices and expert users?
User/task considerations (for categorizing and surveying users)

Job/task/Need
- Type of system use
- Frequency of use
- Task importance
- Task structure
- Social interactions
- Training
- Turnover rate
- Job category
- Lifestyle

Important
User/task considerations (for categorizing and surveying users)

Psychological Characteristics re: job/task
- Attitude
- Motivation
- Patience
- Expectations
- Stress level
- Cognitive style
User/task considerations (for categorizing and surveying users)

Physical characteristics

- Age
- Gender
- Handedness
- Disabilities
Many researchers have studied the speed at which people can perform using various communication methods. The following, as summarized by Bailey (2000), have been found to be typical interaction speeds for various tasks. These speeds are also summarized in Table 1.6.

**Table 1.6: Average Human Interaction Speeds**

<table>
<thead>
<tr>
<th>Activity</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reading (prose text)</td>
<td>250–300 words per minute</td>
</tr>
<tr>
<td>Proofreading text on paper</td>
<td>200 words per minute</td>
</tr>
<tr>
<td>Proofreading text on a monitor</td>
<td>180 words per minute</td>
</tr>
<tr>
<td>Listening</td>
<td>150–160 words per minute</td>
</tr>
<tr>
<td>Speaking to a computer</td>
<td>105 words per minute</td>
</tr>
<tr>
<td>After recognition corrections</td>
<td>25 words per minute</td>
</tr>
<tr>
<td><strong>Keying: Typewriter</strong></td>
<td></td>
</tr>
<tr>
<td>Fast typist</td>
<td>150 words per minute and higher</td>
</tr>
<tr>
<td>Average typist</td>
<td>60–70 words per minute</td>
</tr>
<tr>
<td><strong>Computer</strong></td>
<td></td>
</tr>
<tr>
<td>Transcription</td>
<td>33 words per minute</td>
</tr>
<tr>
<td>Composition</td>
<td>19 words per minute</td>
</tr>
<tr>
<td><strong>Two Finger Typists</strong></td>
<td></td>
</tr>
<tr>
<td>Memorized text</td>
<td>37 words per minute</td>
</tr>
<tr>
<td>Copying text</td>
<td>27 words per minute</td>
</tr>
<tr>
<td><strong>Hand Printing</strong></td>
<td></td>
</tr>
<tr>
<td>Memorized text</td>
<td>31 words per minute</td>
</tr>
<tr>
<td>Copying text</td>
<td>22 words per minute</td>
</tr>
</tbody>
</table>

FROM: UI and Usability Req. - Use cases

One of the most powerful methods. *Must do!*

- Define actors in the application
- Define user skill level
- Describe user tasks and errors
- Know frequency and importance of tasks
- Describe user environment and context
- Describe or use data items from data glossary – be consistent

Methods for gathering information:

- Ask users (single or team)
- Observe users in their actual environment
- Look at user documentation or any other information
- Surveys, focus groups
- Ask your own marketing, sales, support, client reps
- Read WWW, product reviews
- Ask your friends
UI and Usability Requirements - Use Cases >> Task analysis

• Task name, description and current duration
• User skill level, experience and role
• Work steps including usage of other systems (printers, faxes...)
• System features, GUI screens currently used
• Difficulties and errors in using current system
• Current systems performance and reliability
• Current user satisfaction and “pain points”
• Follow-up questions
• Sample inputs, screens etc.

• Good to compare current and future (with your system) use cases in terms of number of steps, time to complete etc.
Example Use Case – Paper Form Entry – structured version

- Users skills: high school. Minimal training for the task (1 hour)
- Work setup: desk. Cluttered with a lot of paper, and not enough space
- Form (show picture)
- Work steps: Take paper from the pile, places at the middle of the table and copies entries into the form.
- Difficulties and errors: Paper is often lost. Drops from the table. Papers get mixed up.
- Performance: 10 minutes per form.
- Reliability: 5 errors per form on average. 10% forms are lost
- User satisfaction: Very frustrated. Tires easily. High staff turnover
Example Use Case – e-TAT – teamwork assessment tool – free text

John is the instructor of SE class. He has already adopted the recommended learning materials and methods. However, he also needs to assess and monitor student teamwork during the class. John first sets up the recommended class tools (for collaboration and SW development) and forms the student teams using the learning and teaching material instructions, resources and tools. He downloads, installs and familiarizes himself with e-TAT using recommended tutorials and documentation. Using the admin module of e-TAT John initializes the student teams.

Based on his class objectives, he modifies and customizes the questionnaires and final presentation data within the options provided by e-TAT.

(BOLD items should be defined in data glossary)
Summary

- Know who your users are, what are their skills, motivation and TASKS they want to do
- Use this to guide the usability designs
- Start with use cases
API usability and requirements for the design of good APIs

- APIs are a form of human/machine UI – user is the programmer ➔ API usability means make it easy for programmers to use your APIs

- Good articles
  - [http://www.davekoelle.com/api_usability.html](http://www.davekoelle.com/api_usability.html)

http://www.drdobbs.com/184405654;jsessionid=WIZ2G1PICRYATQE1GHOSKHWATMY32JVN
API usability – what is it

- How well are APIs designed – do they provide all the functions, are the variables intuitive and useful etc. – use UCD for the design, think as a user (who is the user here?)
- Documentation?
- Easy to download and link to
- Are simple application examples provided?
- Do they run on all necessary platforms?
- Managing API changes vs. legacy installations
- Test by asking other programmers to use your APIs and docs
Practical suggestions on GUI and usability

Implement and follow UCD!

- Develop good use cases and data dictionary first
- Use the above to drive the UI design
  - Cover all use cases
  - Use the same functions names and text as in data dictionary
- Do not add functions not required by specs
- Start with mockups and test them using use cases BEFORE the implementation
  - List major functions needed to implement use cases
  - Sort by priority, importance, frequency of usage
  - Make it easy to access/use those most important
Practical suggestions on GUI and usability

- Adhere to GUI design principles, focus on:
  - Simplicity
  - Good layout (alignment, grouping, font) and workflow
  - Critical importance of UI text and messaging
  - Black and white is OK!

- Use design patterns and widgets from reputable companies for basic functions (menus, clock, calendar...)
TEXT in GUI

- It is critical
- Use action oriented mode “To do X ....”
- In menus/lists be consistent: either use nouns or verbs but do not mix
- Requires great care –every sentence has to be checked
Practical suggestions on GUI and usability

- Home page: tell me about what you offer, who you are, how to learn more, and how to start
- Each page: have title, a few lines explaining the purpose
- Arrange functions by probability/importance of usage – left to right, top to bottom
- Text: action oriented: “To start login here ...”, use defined terms
- Always make it easy to: go back, go home, know where you are
- Avoid scrolling, especially horizontal
  - Do not put major buttons at the bottom which might then be obscured due to the need for vertical scrolling
- Key action buttons (SAVE, APPLY, CONTINUE) to be on the right close to the field they refer to
- CANCEL, RESET to the left and away from the key buttons
Practical suggestions on GUI and usability

- Menu items: make default menu item: “chose one” or “select” or “---”---” so it can be checked to make sure users actually chose one
- Group similar things together by space, make same font, put a frame around them:
  - Address
  - Personal data
  - Account data
- Alignment: fields aligned to the left
- Give examples for entry format if entry is complex (e.g. phone, soc. Security, credit card number)
Search GUI

- Make it simple (text entry preferred to complex Boolean)
- Make it easy to browse (browsing is preferred to search...)
- Make it clear what domain is searched (internal DB or external WWW, WWW news or WWW video, books vs. movies...) – can be pull down next to search field
- Make it easy to view results efficiently w/o full download of media (provide samples, key frames, audio samples, text snippets..) e.g., United Airline email notification sample
- Let user know total items found, what items are currently displayed, have PREVIOUS and NEXT buttons to go through the results list
- Have “back to search” link leading to the search screen with previously entered terms so they can be revised

Review: www.amazon.com
Practical suggestions on GUI and usability

- Browser compatibility: Not only focus on a few latest browsers, but also previous ones
- Test for proper resolution including netbooks and handheld
- Cross browser issues – not trivial
  - Do not use esoteric browser specific code – program for portability among browsers
  - Check WWW resources for guidance and well tested and portable widgets
    - Browser compatibility Tutorial
    - 7 Fresh and Simple Ways to Test Cross-Browser Compatibility (February 2009)
      http://freelancefolder.com/7-fresh-and-simple-ways-to-test-cross-browser-compatibility/
Practical GUI development process as per UCD

- Develop use cases
- Define and prioritize functions
- Create paper B&W mockups (NO programming)
  - Review mockups with internal users or domain experts and developers (internal focus groups) – revise revise revise
- Create horizontal prototype – clickable UI but no back end
  - Review mockups with users or domain experts and developers (interval focus groups) – revise revise revise
- Start coding
  - Get feedback, revise
- Show to wider audience including limited external users
  - Get feedback, revise
Usability testing and focus groups

- **Usability testing**: typical users given product and test cases; users test the system and answer questionnaire
  - Usually done toward the end of the cycle
  - Good for verification but gives little design guidance how to improve

- **Focus groups (internal or external)** – combine usability testing AND design improvements: preferred in early stages
  - 3-8 people in the room observe the product together and try use cases
  - Discuss problems, come up with better design
  - Book keeping records the decisions /tasks (usually a lot of small details)
  - Changes made – keep the list of tasks and what was changed
  - Repeat and iterate (internally then externally)
How to give GUI feedback - format

- Use text to describe use case and the problem (record the browser, OS etc.). Optionally suggest improvement.
- Combine with graphical representation (annotate screen shots using Adobe PDF postit or text on PPT).
I suggest having a page title like “Join the project: register as Citizen Scientist”. This way we also introduce the key term “citizen scientist” which is totally not visible now (see more on this).

I would omit this text. Stay focused on Registration, by now We explained what Is this about (this should be In ABOUT).

I would start by saying” Get involved, become Citizen Scientist by registering On this page.

I suggest: Create an account and become Citizen Scientist.

Actual Example from www. zombeewatch.org development

1. Create an account
2. Read the tutorial
3. Submit data

- Email:
- Password:
- Confirm Password:
- Screen Name:
- Bee Keeper: I accept the terms of use

Are we checking For uniqueness?

Align check buttons

“Your screen name”

Pop up explain
After the change

ZomBee Watch
A citizen science project tracking the honey bee parasite *Apocarphalos boralis*

1. Create an account
   - Join the project and become citizen scientist.
   - You will help us by sending information we can share with others.
   - To join, first create an account

2. Read the tutorial

3. Complete registration

Create an account
Create an account and become a citizen scientist.

Email: 
Use email for logging in.

Your Screen Name: 
Screen name has to be unique.

Password: 

Confirm Password: 

Type text below: 

Beekeeper: I am a beekeeper.

Terms of use: I accept the terms of use

Register!
GUI QA

- QA = find bugs vs. specs (different from usability testing e.g. determine how easy is it to use)
- QA done by traditional QA departments – passing QA does NOT mean the app is usable
- GUI QA is time intensive: try all buttons, all functions, test all paths of usage, test for all browsers, resolutions and users – drive by use cases
- QA Automation tools for GUI QA exist – use and learn them
  - [http://seleniumhq.org/](http://seleniumhq.org/)
  - [http://www.froglogic.com/?gclid=CPz6oqysuaoCFcgZQgodtx0X5Q](http://www.froglogic.com/?gclid=CPz6oqysuaoCFcgZQgodtx0X5Q)
Review Questions

- What are GUI design principles vs. GUI patterns vs. widgets (familiarize yourself with major libraries)
- Know basic GUI design principles
- Differences between WWW and mobile platforms
- Usability testing vs. focus groups – what they are and when to use which
Review Questions

- What is usability vs. GUI
- Describe your experience of satisfaction or dissatisfaction with SW usability and UI
- What are challenges in SW development for UI
- What is SW engineering method of choice for GUI. Describe its components and key properties that distinguish it from other SW engineering methods
- Compare classic native GUI (i.e. Windows) with WWW UI for: user conceptual model, typical usage and tasks, presentation elements and visual style, navigation and context.
Review Questions

- Sketch a typical layout of WWW pages today i.e. Amazon, Yahoo). What are commonalities?
- List at least 10 GUI design principles. Be able to describe each. Think of good or bad examples for each principle.
- What are your top 5 GUI design principles you would like to follow?
- Describe Fits law. What design guidelines come from this?
- What are considerations for characterizing users?
- Types of users vs. their skill level
- What are market leading tools for GUI QA testing?
- Define API usability and its importance