HY 351: Ανάλυση και Σχεδίαση Πληροφοριακών Συστημάτων
CS 351: Information Systems Analysis and Design

User Interface Design

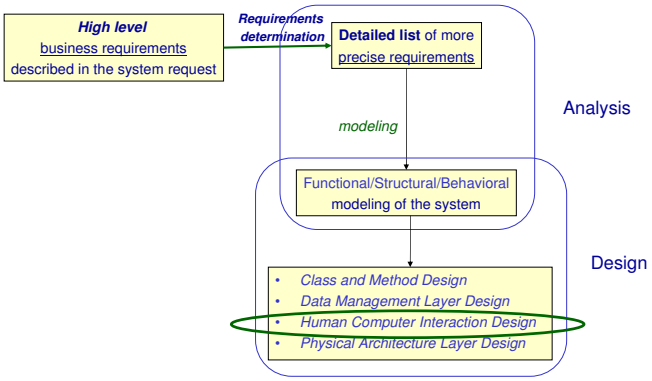
Lecture : 17
Date : 1 -12-2005

Yannis Tzitzikas
University of Crete, Fall 2005

Outline

- What is HCI ?
  - Why the User Interface is an important part of an IS?
- General Principles of UI Design
- Input / Output / Navigation mechanisms
- The UI Design Process
  - How we evaluate a UI
- UI Design and UML (the UX modeling language)

From Analysis Models to Design Models



What is HCI?
Why the UI is an important part of an Information System?

HCI: The dialog between the user and the system

- System Interface:
  - defines how systems exchange information with other systems
- User interface:
  - defines how the system will interact with external entities

A "good" UI:

- *Can reduce training costs*
- *Can increase the productivity of its users*
- *Can prevent user errors*
- *Can satisfy the users*

Why the UI is an important part of an IS?
Can reduce training costs

- 10 employees
- 2 new applications per year
- 2 days saved from training time

= 10 x 2 x 2 = 40 days savings per year (2 person months)

2 months x 1500 = ***3.000 Euros / year***

Why the UI is an important part of an IS?
Can increase the productivity of its users

- 10 employees
- 230 working days per year
- 100 screens per day
- 10 sec per screen (savings)

= 10 x 230 x 100 x 10 /3600 = 638 hours per year (savings)
= 638 / 8 = 79 days / 20 ≈ 4 months per year (savings)

4 months x 1500 = ***6.000 Euros / year***

126 SM

Why the UI is an important part of an IS?
Can prevent user errors

- 500 users
- 20 errors per year
- 15 minutes per error

= 500 x 20 x 15 / 60 = 2500 hours

= 2500 hours / 8 (=312 days) / 20 ≈ 15 months lost per year

15 months x 1500 = ***22.500 Euros / year***

Why the UI is an important part of an IS?
Can satisfy users (*and … sell the system!*)

It is important to remember that to the end user
*"the user interface **IS** the system",*

- Users tend to reject (even sabotage) system with UI they don't like
- A good UI can sell the systems !
- A poor UI can ruin a system that is outstanding in all other aspects

Some statistics

- UI is 47% - 60% of the total code
- The GUI is minimally 29% of the software development project budget
- The GUI may take as much 40% of the development effort

Consequences of poor UIs

Real cases:

- A $3 million application for an insurance company to be used by independent agents to support them in selling their company's products. However, agents refused to use the application because the system was "un-learnable" and "unusable".
- In a customer service organization, training on the system took 6 months, but employees typically stayed only 18 months in that department.
- Extensive and expensive functionality in a Human Resources system was not used because users forgot how to access it one week after training.

Why there are poor UIs?

- Examples of poor user interfaces in widely used mechanical system:
  - e.g. VCRs, Photocopies, Faxes, ..
- Why they are poor ?
  - Inadequate training of developers
    - diverse knowledge required to design interfaces
    - user interface specialists are rarely involved
  - Ignorance by software engineers of usability and how to measure it
  - The Multidisciplinary Nature of HCI

Multidisciplinary Nature of HCI

Human side:
- cognitive psychology
- ergonomics and human factors
- sociology and anthropology
- linguistics
- communication theory
- social and organizational psychology
- graphic and industrial design

Machine side:
- computer science
- engineering
- computer graphics
- operating systems
- programming languages
- software engineering
- development environments
- artificial intelligence

Human Characteristics

- Limited short-term memory
  - People can instantaneously remember about 7 items of information. If you present more than this, they are more liable to make mistakes.
- People make mistakes
  - When people make mistakes and systems go wrong, inappropriate alarms and messages can increase stress and hence the likelihood of more mistakes.
- People are different
  - People have a wide range of physical capabilities. Designers should not just design for their own capabilities.
- People have different interaction preferences
  - Some like pictures, some like text.

Human Characteristics and .. their implications on UIs

Some human characteristics:

- People don't like reading manuals
- People use prior learning to support new learning
- People are always building models of their world

Implications

- build interfaces that allow users to learn by using the interface
- build interfaces that suggest correct models
- build interfaces that rely on prior learning

So we can safely assume that users already know the "semantics" of the icons used in MS Windows/Office, e.g.: 

Principles of UI Design

Design Principles

User familiarity

- The interface should be based on user-oriented terms and concepts rather than computer concepts.
  - E.g.: in an office system use concepts such as letters, documents, folders etc. rather than directories, file identifiers, etc.

Consistency

- The system should display an appropriate level of consistency. Commands and menus should have the same format, command punctuation should be similar, etc.

Minimal surprise

- If a command operates in a known way, the user should be able to predict the operation of comparable commands

Recoverability

- The system should provide some resilience to user errors and allow the user to recover from errors. This might include an undo facility, confirmation of destructive actions, 'soft' deletes, etc.

User guidance

- Some user guidance such as help systems, on-line manuals, etc. should be supplied

User diversity

- Interaction facilities for different types of user should be supported. For example, some users have seeing difficulties and so larger text should be available

Features of a Good Design

- Has **affordances**: makes each operation visible
  - e.g. buttons that indicate that they are to be pressed, scrollbars, glass is for looking through or for breaking, stairs are for climbing, ...
- Offers obvious **mappings**: makes the relationship between the actual action of the device and the action of the user obvious
  - good example: the presentation of the font to be selected in MSWord
  - bad example: function keys
- Provides **feedback** on the user's actions
  - good example: *WYSIWYG* (*What You See Is What You Get*) e.g. Word
  - bad example: latex
- Provides a good **mental model** of the underlying behavior of the device
- Provides **forcing functions** : prevents a user from making bad errors
  - good example: inactive menu items
  - bad example: command line interfaces (the user can issue any command)
- Supports **automatic learning**: offers consistencies and practice that help the user acquire interface skills (based on repetitive patterns).
  - good example: a set of screens where the locations of the menu items are the same
  - bad example: a set of screens where the locations of the menu items are different

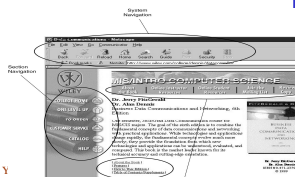
Principles for User Interface Design

- **Layout**
- **Content awareness**
- **Aesthetics**
- **User experience**
- **Consistency**
- **Minimal user effort**

Principles> Layout Concepts

- "Similar" areas should be grouped together (the user movement should be minimized)
- Areas and information should minimize user movement from one to another
- Ideally, areas will remain consistent in
  - Size
  - Shape
  - Placement for entering data
  - Reports presenting retrieved data

- The screen is often divided into three boxes
  - Navigation area (top)
  - Status area (bottom)
  - Work area (middle)
- Information can be presented in multiple areas



Principles> Content Awareness

- All interfaces should have titles
- Menus should show
  - where you are
  - where you came from to get there
- It should be clear what information is within each area
- Fields and field labels should be selected carefully
- Use dates and version numbers to aid system users

Principles>Aesthetics

- Interfaces need to be functional and inviting to use
- Avoid squeezing in too much, particularly for novice users
- Design text carefully
  - Be aware of font and size
  - Avoid using all capital letters
- Colors and patterns should be used carefully
  - Test quality of colors by trying the interface on a black/white monitor
  - Use colors to separate or categorize items

Principles> User Experience

- How easy is the program to learn?
- How easy is the program to use for the expert?
- Consider adding shortcuts for the expert
- Where there is low employee turnover, some training can lessen the impact of less precise interfaces

Principles> Consistency

- Enables users to predict what will happen
- Reduces learning curve
- Considers items within an application and across applications
- Pertains to many different levels
  - Navigation controls
  - Terminology
  - Report and form design

Principles>Minimize Effort

The *"3 clicks rule"*:

Users should be able to go from the start or main menu of a system to the information or action they want in no more than three mouse clicks or three keystrokes

The 3 fundamental parts of a UI:

Input mechanism: defines the way the system captures information

Output mechanism: defines the way the system provides information to users or other systems

Navigation mechanism: provides the way for users to tell the system what to do

Input Mechanism

Input mechanism

Input mechanisms allow the entry of data in the system.

- Data capture
  - involves the identification of new data to be inserted in an information system
    - e.g. a photo, a bar code
- Data entry
  - the process of translating the source document into a machine readable format
    - e.g. digitazing the photo
- Data input
  - the actual entry of data (already in machine-readable from) in the computer

Online versus Batch Processing

- **Online processing** immediately records the transaction in the appropriate database
- **Batch processing** collects inputs over time and enters them into the system at one time in a batch
  - Batch processing simplifies data communications and other processes, but means that inventory and other reports are not accurate in real time

Input Design

Input design means designing the screens used to enter the information as well as any forms on which users write or type information.

The goal is to simply and easily capture accurate information for the system

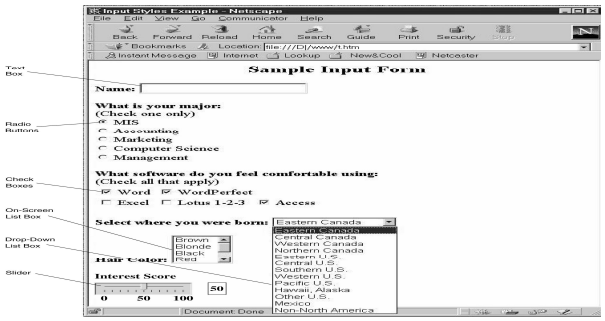
Principles

- Capture Data at the source
  - Reduces duplicate work and processing time
  - Decreases cost and probability of error
- Minimize keystrokes
  - Never ask for information that can be obtained in another way
  - List selection is more efficient than entering information
  - Use default values where possible
- Validate input
  - Check the completeness, format, range, consistency of data entry
  - Data errors can be reduced dramatically because of data validation processes

Input Devices

- Input devices:
  - keyboards, mouse
  - bar code readers
  - optical character recognition
  - magnetic stripe readers
  - smart cards
  - scanners
  - cameras + software

Types of Input Boxes



Output Design

Output Design

Types of Outputs

- External outputs
  - leave the system permanently
    - airline tickets, boarding passes, bank transaction receipt
- Turnaround outputs
  - leave and later reenter the system
    - invoices, purchase orders
- Internal outputs
  - never leave the system (useful for monitoring and management purposes, e.g. internal reports, summary reports, etc).

Basic Principles

- Understand report usage
  - Frequency?
  - Real-time or batch reports?
- Manage information load
  - All needed information, no more
- Minimize bias
  - e.g. scales in bar charts should be defined carefully

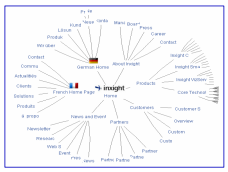
Output Design>More advanced building blocks

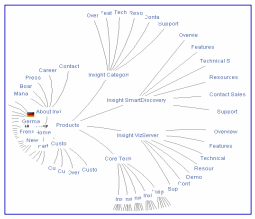
Advanced visualization techniques are useful for providing overview data which are very important for taking strategic decisions.

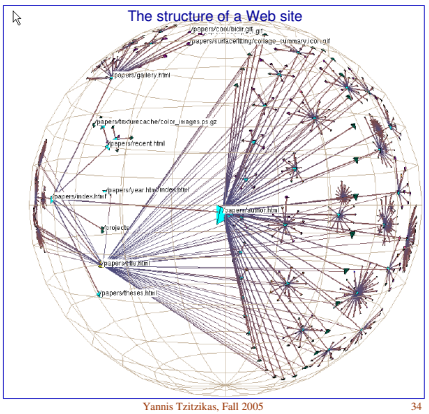
Examples:

- Bar charts (histograms) and pie charts
- Maps (with clickable spots)
- Three/Graph-based layouts

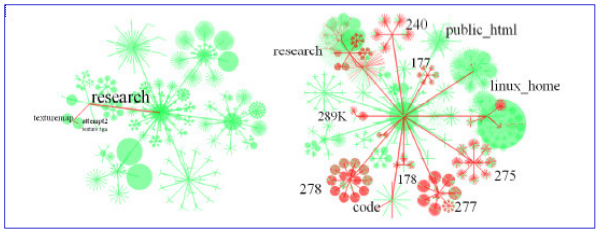
Advanced Visualisation Techniques
Overviews: Hyperbolic Trees

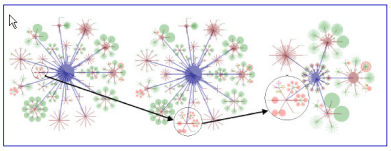




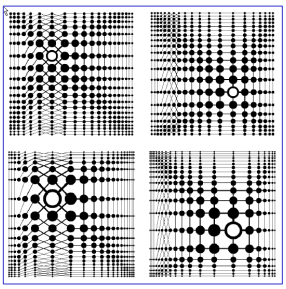


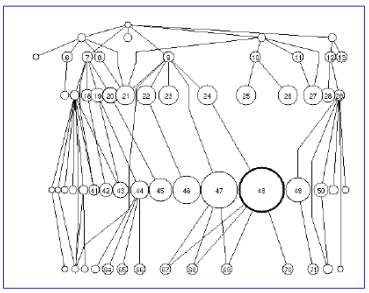
Advanced Visualisation Techniques
Rings





Advanced Visualisation Techniques
Fisheye views





Advanced Visualisation Techniques

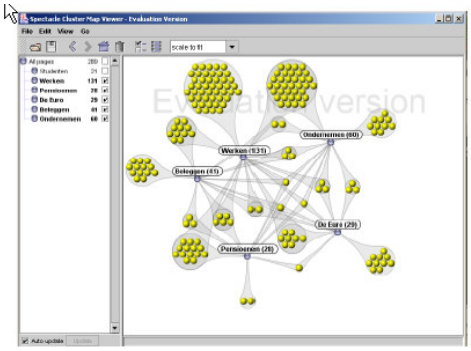
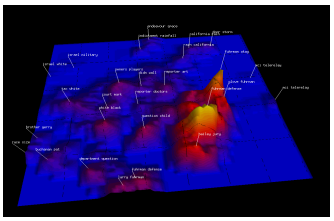


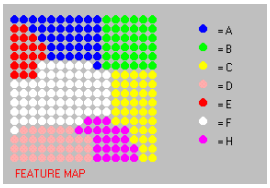
Figure 1: Spectacle

Advanced Visualisation Techniques
For clustered data

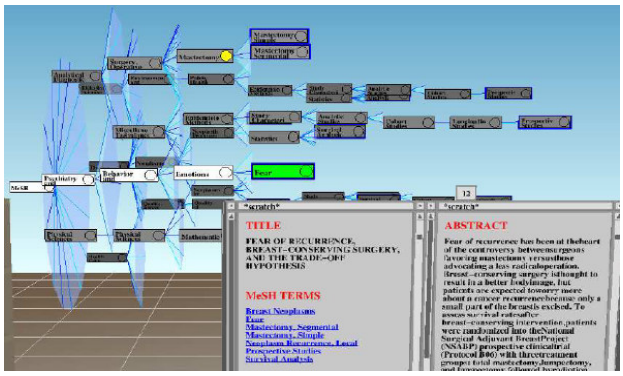
Three-dimensional overview based on clustering (**Themescape**)



Kohonen's feature map



Advanced Visualisation Techniques
...



Navigation Mechanism

Navigation mechanism

- Hardware
  - Keyboard, mouse, touch screen,
  - Voice recognition systems, cameras, eye movement tracking, …
- Software
  - Command line
  - Menus
  - Direct manipulation (e.g. resize the window, … right click

Principles

- Assume that users have not read the manual, have not attended training and do not have any external help
- All controls should be clear and understandable and placed in an intuitive location on the screen.
- Prevent mistakes
  - Never display a command that cannot be used (inactive commands)
  - Ask for confirmation for functions that are difficult or impossible to undo
- Simplify recovery from mistakes
- Use consistent grammar order (e.g. object-action order)

Navigation mechanism: Menus and Messages

Menus

Types of Menus

- Menu bar, Drop-down menu, Pop-up menu, Tab menu,Toolbar, Image map

Principles

- Avoid having more than 8 items per menu.
- If more then group the items of the menu.
- Group on the basis of objects (e.g. File, Table, Window in MS Office appls)
- Group on the basis of task (e.g. Edit, Insert, Format)
  - Note: the menus of MS Office appls follow a mixed approach
- Support shortcut keys (or hot keys) for common actions (e.g. Alt-F for search)
- The user should be able to reach (from the start menu) the desired action with no more than 3 mouse clicks.

Messages

Types of Messages

- Error messages
- Confirmation messages
- Acknowledgement messages
- Delay messages
- Help messages

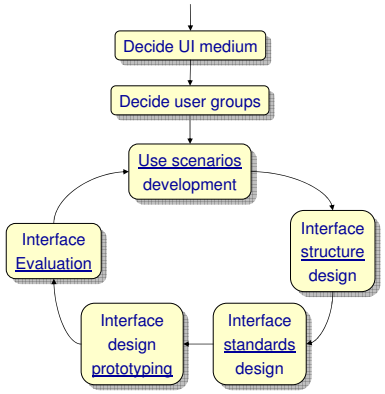
Principles

- Should be clear, concise, and complete
- Should be grammatically correct and free of jargon and abbreviations
- Avoid negatives and humor

Designing User Interfaces

Designing User Interfaces
A Seven Step Process

1 Decide UI medium
2. Decide user groups
3. Use scenario development
4 Interface structure design
5 interface standards design
6 Interface design prototyping
7 Interface Evaluation



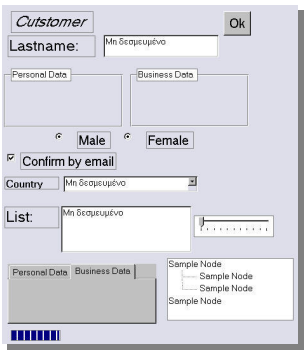
UI Design Process
1. Decide UI medium

Design Process
1. Decide UI medium (hardware)

- Monitors
  - display area
  - character-based
    - old terminals, old mobile phones
    - gadgets (e.g. an mp3 player)
  - bitmap-based (analysis, bit, …)
  - touch screen monitors
- Hardware for virtual reality, …
- …

The basic building blocks of a GUI

- Buttons (text labelled or with icons)
- Labels
- text boxes
- composite boxes
- scroll bars
- lists
- radio boxes
- check boxes
- menus
- pop-up windows
- status bars
- …



UI Design Process
2. Decide User Groups

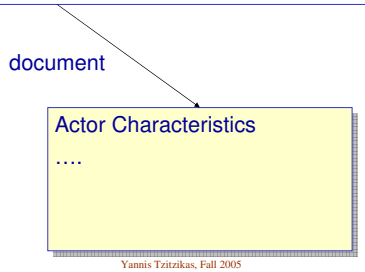
Design Process
2. Decide User Groups

- We may have different groups of users
- Each one may require it's own interface
  - e.g. from web sites
    - *for users*
    - *for visitors*
    - *for authors*
  - e.g. the users of one department of an organization

- See Actors from the Use Cases and the Use Case Diagrams
- Sometimes we may have to define more than one user group for each actor
  - E.g. novice users vs experienced users
  - Characteristics / knowledge of users

For each actor/user group

- Describe user's average level of domain knowledge
- General level of computer experience
- Approximate number of users represented by an actor.
- How frequently they will use the system?



UI Design Process
3. Use scenario development

Design Process
3. Use Scenario Development

- A **use scenario** is an outline of the steps that the users perform to accomplish a task
- A use scenario is one path through an essential use case

- We describe use scenarios by a simple narrative tied to the related use case.
- We usually document the 2 to 3 most common use scenarios per use case just to enable UI design.
  - Documenting every possible use scenario would be very time consuming

UI Design Process
4. Interface Structure Design

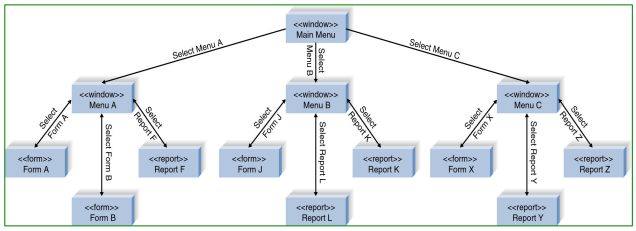
Design Process
4. Interface Structure Design

Use cases and use scenarios do not specify any user interface details.

Here we have to specify the basic components of the interface and how they will work together to provide the desired functionality to users.

Window navigation diagram (WND)
- Shows how all screens, forms, and reports are related
- Shows how user moves from one to another
- Like a state diagram for the user interface
  - **Boxes** represent components
  - **Arrows** represent transitions
  - **Stereotypes** show interface type
- Differences between State diagrams and WNDs
  - State diagrams describe the states of an object
  - WNDs describe the states of the UI

Design Process
Example of a WND



Single-headed arrow: a return to the calling state is not required
Double-headed arrow: return is required

We will also discuss UX Modeling (next tutorial)

UI Design Process
5. Interface Standards Design

Design Process
5. Interface Standards Design

- Interface standards are the basic elements that are common across individual screens, forms, and reports within the application
- There may be more than one interface standard in a system.
  - E.g. one for Web screens, one for Reports and one for input forms

- Interface templates
  - An interface template defines the general appearance of all screens, e.g. the basic layout, the color scheme, windows "replacing" policy (replace/cascade), etc.
- Interface metaphor
  - The fundamental interface metaphor should be defined.
  - The metaphor (which originates from the real world) helps users to understand the system and predict its behavior
  - Different parts of the system may follow different metaphors
- Interface objects
  - Buttons, icons, button labels form, …

UI Design Process
6. Interface Prototyping

Interface Prototyping

- **Plan** the prototyping process
  - *what needs to be learned?*
  - Identify the highest risk items.
  - Prototype: intended to minimize effort (to build it) and maximize feedback
- **Evaluate** the prototype
  - gather data of how effective the prototype is
  - evaluate the data to identify areas that need further elaboration
  - look for false, missing, useful affordances
  - check for overhead in learning

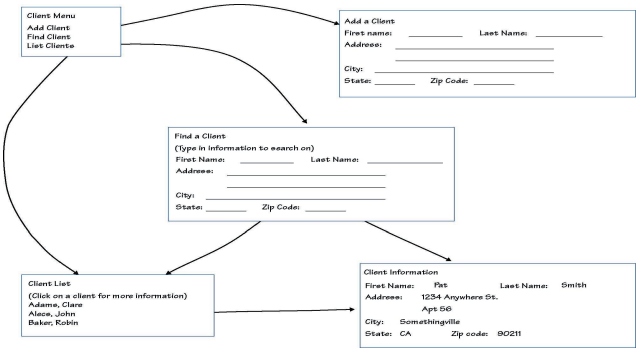
Design Process
Types of prototypes

- On paper
  - A mock-up or simulation of screen, form, or report
  - Storyboarding
- On computer
  - Build a "quick-and-dirty" implementation of the interface in a very high level language or GUI tool to show the user how the UI will look like
  - Examples
    - HTML prototyping
      - The user can click
    - PL-based prototyping
      - E.g. in Visual Basic

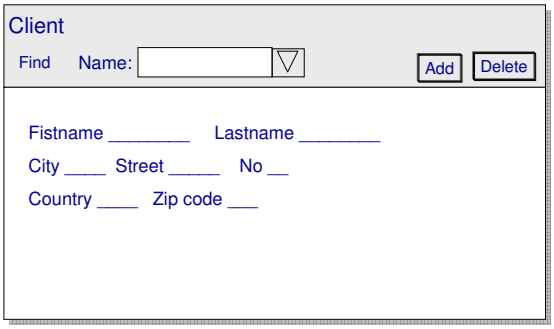
- Prototypes offer only approximate look-and-feel, can't be used to assess response times

Mock-up: φυσικού μεγέθους μοντέλο μηχανής προς διδασκαλία

Design Process
Storyboard Example



Design Process
An alternative …. "storyboard"



UI Design Process
7. Interface Evaluation

Design Process
7. Interface Evaluation

- Ideally the UI should be evaluated at design time
  - (and not at the end of the project)
- The objective is to understand how to improve the UI before implementing it
- As UI design is quite subjective, it is important to have many (at least 10) potential users to evaluate it.

- **Evaluate** the prototype
  - gather data of how effective the prototype is
  - evaluate the data to identify are as that need further elaboration
  - look for false, missing, useful affordances
  - check for overhead in learning

Design Process
Interface Evaluation Methods

- **Heuristic evaluation**
  - Evaluate the UI by comparing it to a set of heuristics or principles for UI design.
  - These heuristics/principles usually have the form of a checklist
  - At least 3 persons from the team individually examine the UI and check whether is conforms to the checklist.
- **Walkthrough evaluation**
  - A meeting with the users of the system is scheduled where the team simulates the functionality of the UI (with the aid of the storyboard or the prototype). Users identify problems and suggest improvements.
- **Interactive evaluation**
  - The users themselves try out the UI. A prototype is needed for this case. Users identify problems and suggest improvements.

An Indicative Usability Checklist (1/2)

Heuristic evaluation is conducted by using these heuristics to find usability errors.

- Visibility of system status
  User should always know what is going on and the feedback time of the system should be reasonable.
- Match between system and the real world
  The language of the system should meet users language rather than system depended language. When designing the system from the user point of view, the users must be identified and also the language they will use in the system must be found. Language should also follow real world conventions and phrases.
- User control and freedom
  Users make mistakes and system must offer "emergency exits" to get out of unwanted state as easy as possible. Also undo and redo should be supported.
- Consistency and standards
  Use same words and phrases for same things through the system. For same action there should not be two or more different words. User interface should follow normal platform based conventions.
- Error prevention
  To design the system error resilience and prevent problems from occurring in the first place is more important than good error messages.

An Indicative Usability Checklist (2/2)

- Recognition rather than recall
  Keep things visible. The user should never need to remember information from the other view to be able to execute other view's operation. The important information should be retrievable when ever needed.
- Flexibility and efficiency of use
  Accelerations. In some cases the system automatically allows users to use some acceleration to speed up the interaction and these accelerations should never been disabled. Also the system should allow the user to tailor accelerations. The accelerations allow more experienced users to be more efficient and the system still can support novice users to execute required tasks.
- Aesthetic and minimalist design
  View should contain only relevant and usable information.
- Help users recognise, diagnose, and recover from errors
  Error messages should contain plain language, not error codes, to indicate the error. Also suggestion on how to proceed should be provided.
- Help and documentation
  The user should have easy access to the help (online) and to the documentation. This heuristic is left out or used very lightly in this project. The main reason is that evaluation will be done lightly and the evaluations will happen in the part of the project where help and documentation is not available or has just started.

Interface Evaluation Methods
Formal Usability Testing

- It is a formal procedure that requires having a prototype.
- It requires a lot of preparation and it is a very expensive procedure (if done very formally).
- Users individually try out the system. They are given specific tasks (e.g. use scenarios) to accomplish. At the beginning, they are given specific instructions (amongst others, this allow estimating how much training will be required for the new system).
- The time the user spent to accomplish his/her tasks, and the "quality" (correctness or …) of the outcome is measured. User can also fill some questionnaires at the end of the evaluation.
- Sometimes the testing takes place in a lab equipped with video cameras and/or with software that records each and every keystroke and mouse operation of the user.
- Formal usability testing is expensive and time consuming therefore it is usually done only with commercial (generic) software products.

Rating the (usability) problems

All the founded problems can be rated to indicate problem severity. The rating can be designed purely for the project point of view or can adopt some used practice.

- Rating the severity of problems using a single scale
  - 0: Not a usability problem
  - 1: Cosmetic problem. Will be fixed only if there is extra time.
  - 2: Minor problem. Fixing this is low priority.
  - 3: Major problem. High priority and very important to fix.
  - 4: Critical. Prevent using the system. Must be fixed before version release.
- Rating the severity of problems using a three scales:
  - Frequency. How often the problem occurs
  - Impact. How easy it is for user to overcome the problem.
  - Persistence. Can user overcome the problem when he knows about it or does it occur repeatedly

Designing User Interfaces with UML

(The User-Experience (UX) Modeling Language)

Designing User Interfaces with UML

Structural and Behavioral modeling can be applied for modeling and designing a User Interface

- Structural elements
  - screens, forms, …
- Behavioral elements
  - states
  - actions
    - user actions
    - environmental actions

The User-Experience (UX) Modeling Language

- It captures concepts used for representing the engineering aspects of screen-based UIs concerned with:
  - The **screens** the user sees
  - The **actions** the user can perform on these screens
  - They **dynamic data** the system must display on screens
  - The **data the user puts** on the screen
  - The **decomposition of screens** into smaller areas that should be managed separately from other areas
  - **Screen transitions** caused by **user's actions** and **application's reactions**
- The engineering aspects of UIs are not concerned with colors, fonts, layout, etc.

- It is defined as a UML Profile. A UML profile is a collection of metamodel elements, which have been customized for a specific domain.
- UML metamodel elements can have their semantics customized using *stereotypes, tagged value definitions* and *constraints*

UX: Stereotypes

Name	Extends	Description
screen	class	Represents the final visualized UI presented to the user of the sytem
compartment	class	Represents a well-defined region of a screen that is reused by multiple screens.
Input form	class	Represents a group of data fields that can be displayed to and manipulated by the user
form	attribute	Indicates that an attribute (field) on a screen is associated with an input form. See tag values of those attributes
storyboard	package	Designates a package containing a description of a use-case storyboard.

UX: Tagged Value Sets

Attached to the metamodel element Transition

Name	Type	Default Value	Description
System action	String		One or more names of the "application domain" components of the system that will participate in handling the transition

Attached to the metamodel element <<form>> Attribute

Name	Type	Default Value	Description
Editable	Boolean	True	Associated with <<form>> attributes on a screen and indicates that it can be edited by the user
Source Form	String		Associated with <<form>> attributes on a screen. Contains the name of the form that the field comes from
Visibility	String 1: Visible 2: Hidden	1	

Modeling with UX

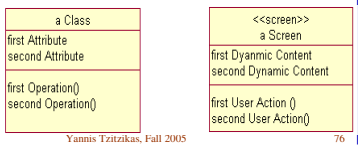
- Primary UX modeling elements
  - Screens
  - Screen compartments
  - Input forms
  - Data structures (I.e. classes with no stereotypes)
- An attribute without a stereotype represents a screen field which output "directly" on a screen
- Operations can be placed on screens, screen compartments and forms
  - An operation represents an action that the user can initiate and the system should handle

OOAD vs UX modeling elements

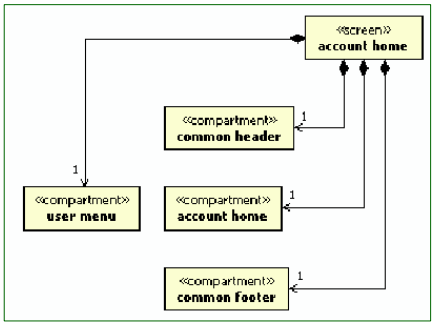
- Mapping between OOAD and UX modeling elements

OOAD Modeling	Ux Modeling
Classes	Screens
Class Diagrams	Navigation maps
Objects	Screen instances
Sequence Diagrams	Screen flow diagrams

- Comparing the characteristics of a class with those of a screen



Examples



Storyboards

- A storyboard is a description of the screen flow of a use case
- A storyboard is represented as a package stereotyped as «storyboard».
- A storyboard information must contain two elements
  - A **class diagram** called "Participants" showing screen-flow dependencies between the UX model elements participating in the storyboard
  - A **state machine** refining the flow of the storyboard and introducing distinct named states of that flow.

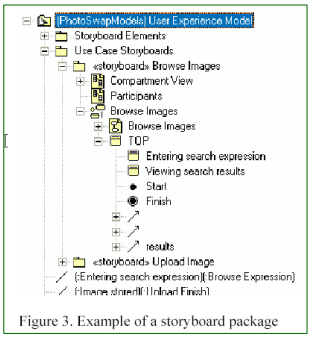


Figure 3. Example of a storyboard package

Examples of a use case storyboard

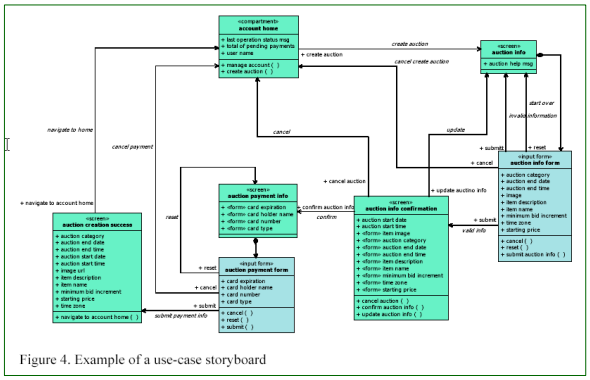


Figure 4. Example of a use-case storyboard

Transforming UX models to code-level models

- It is an instance of the Model-Driven Development
  - i.e. automated development of enterprise applications
- The transformation can be performed
  - by hand
  - automatically be a code generator (e.g. Rational XDE tool)

- XDE has a facility to define and apply model templates called the Pattern Engine. The developer applies a transformation pattern by providing arguments for each template parameter and a target expansion location.
- There are models for Java classes and models for Web elements like Java Server Pages

UX Modeling: Some remarks

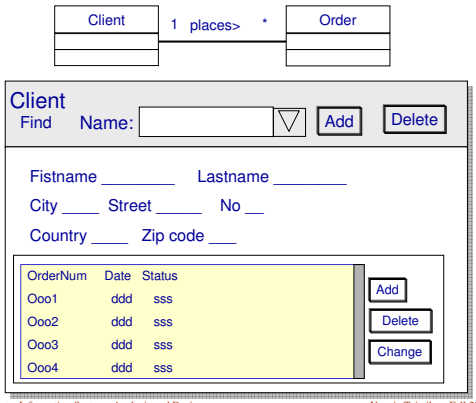
- UX modeling is not required on every project.
- Creating a UX model takes time and effort.
- The benefit of UX modeling outweighs the cost and effort required if the UI is critical to the system's success, e.g. in systems where UI errors might have costly consequences
  - air traffic control systems
  - nuclear power plant systems

Given a class diagram, how you would derive a UI for it?

Some ideas:

- Attribute:String/Int
  - text box
- Attribute {readonly}
  - label
- Association: <<enum>>
  - combo box or radio button
- Association:|end|=1
  - compo box (value the "foreign key")
- Association: |end|>1
  - subform
- Attribute/association derived
  - set appropriately the event "OnUpdate" of the appropriate element(s)
- Method
  - button, menu item

Given a class diagram, how you would derive a UI for it?



UI Design: Summary

- The UI should make the user's work easier and more effective.
- The UI determines the acceptance of an information system by its users
- During the design of a UI we should have in mind
  - cognitive principles about human information processing and memory organization
  - concern for content and context for navigation through activities, aesthetic consideration, assistance for novices and experts, consistency, and minimizing user effort.
- UIs are best developed through prototyping, involving prototype implementations or paper mock-ups
- The design process focuses on user actions, diagramming the structure of the UI, setting up standards (UI templates), and evaluating interface designs.

Reading and References

- **Systems Analysis and Design with UML Version 2.0** (2nd edition) by A. Dennis, B. Haley Wixom, D. Tegarden, Wiley, 2005. CHAPTER 12
- **Software Engineering**, Ian Sommerville, 7th Edition, 2004, Chapter 16
- **Transforming User Experience Model To Presentation Layer Implementations,** Wojtek Kozaczynski, Jim Thario, Rational Software Corporation, USA, 2002
- **User experience storyboards: Building better UIs with RUP, UML, and use cases**, by Jim Heumann, IBM Rational Software Group, 2003