Object-Oriented Software Engineering

Practical Software Development using UML and Java

Chapter 7:

Focusing on Users and Their Tasks

Lecture 10

www.lloseng.com

7.1 User Centered Design

Software development should focus on the needs of users

- Understand your users
- Design software based on an understanding of the users' tasks
- Ensure users are involved in decision making processes
- Design the user interface following guidelines for good usability
- Have users work with and give their feedback about prototypes, on-line help and draft user manuals

The importance of focusing on users

- Reduced training and support costs
- Reduced time to learn the system
- Greater efficiency of use
- Reduced costs by only developing features that are needed
- Reduced costs associated with changing the system later
- Better prioritizing of work for iterative development
- Greater attractiveness of the system, so users will be more willing to buy and use it

7.2 Characteristics of Users

Software engineers must develop an understanding of the users

- Goals for using the system
- Potential patterns of use
- Demographics
- Knowledge of the domain and of computers
- Physical ability
- Psychological traits and emotional feelings

7.3 Basics of User Interface Design

- User interface design should be done in conjunction with other software engineering activities.
- Do use case analysis to help define the tasks that the UI must help the user perform.
- Do *iterative* UI prototyping to address the use cases.
- Results of prototyping will enable you to finalize the requirements.

Usability vs. Utility

Does the system provide the *raw capabilities* to allow the user to achieve their goal?

- This is *utility*.

Does the system allow the user to *learn and to use* the raw capabilities *easily*?

- This is *usability*.

Both utility and usability are essential

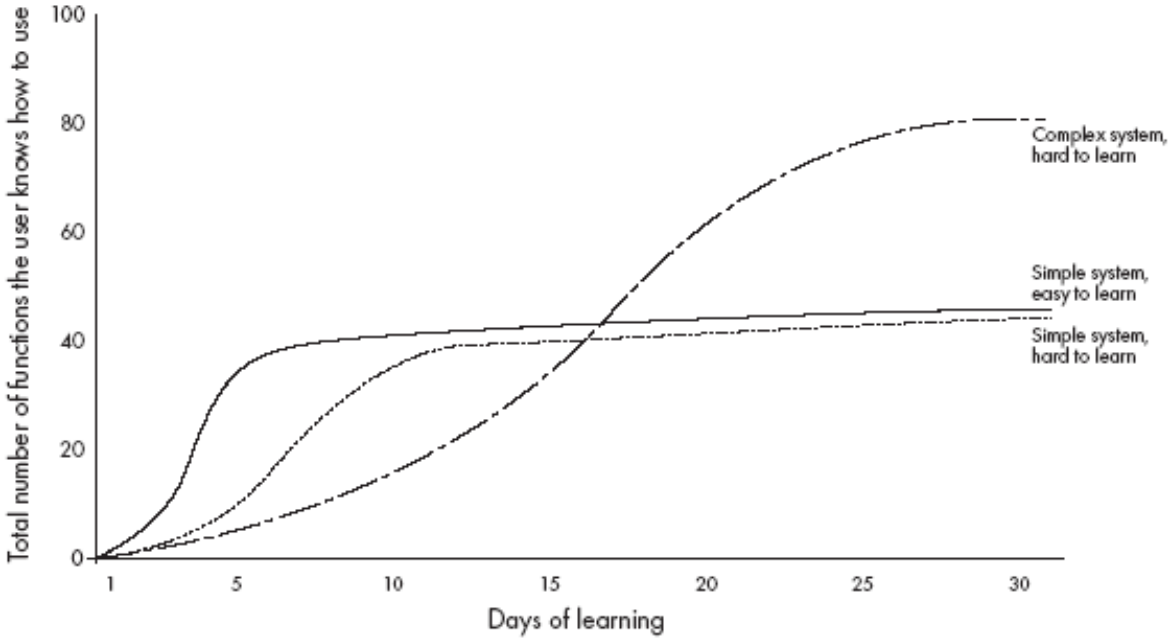
- They must be measured in the context of particular types of users.

Aspects of usability

Usability can be divided into separate aspects:

- Learnability
  - —The speed with which a new user can become proficient with the system.
- Efficiency of use
  - —How fast an expert user can do their work.
- Error handling
  - —The extent to which it prevents the user from making errors, detects errors, and helps to correct errors.
- Acceptability.
  - —The extent to which users *like* the system.

Different learning curves



Some basic terminology of user interface design

- **Dialog**: A specific window with which a user can interact, but which is not the main UI window.
- **Control** or **Widget:** Specific components of a user interface.
- **Affordance**: The set of operations that the user can do at any given point in time.
- **State**: At any stage in the dialog, the system is displaying certain information in certain widgets, and has a certain affordance.
- **Mode**: A situation in which the UI restricts what the user can do.
- **Modal dialog**: A dialog in which the system is in a very restrictive mode.
- **Feedback**: The *response from the system* whenever the user does something, is called feedback.
- **Encoding techniques**. Ways of encoding information so as to communicate it to the user.

7.4 Usability Principles

1. Do not rely *only* on usability guidelines – *always test with users*.

- Usability guidelines have exceptions; you can only be confident that a UI is good if you test it successfully with users.

2: Base UI designs on users' *tasks*.

- Perform use case analysis to structure the UI.

3: Ensure that the sequences of actions to achieve a task are as *simple* as possible.

- Reduce the amount of reading and manipulation the user has to do.
- Ensure the user does not have to navigate anywhere to do subsequent steps of a task.

Usability Principles

4: Ensure that the user always knows what he or she can and should do next.

- Ensure that the user can see *what commands are available* and are not available.
- Make the *most important commands stand out*.

5: Provide good feedback including effective error messages.

- Inform users of the *progress* of operations and of their *location* as they navigate.
- When something goes wrong explain the situation in adequate detail and *help the user to resolve the problem*.

Usability Principles

6: Ensure that the user can always get out, go back or undo an action.

- Ensure that all operations can be *undone*.
- Ensure it is easy to *navigate back* to where the user came from.

7: Ensure that response time is adequate.

- Users are very sensitive to slow response time
  - —They compare your system to others.
- Keep response time less than a second for most operations.
- Warn users of longer delays and inform them of progress.

Usability Principles

8: Use *understandable encoding techniques*.

- Choose encoding techniques with care.
- Use labels to ensure all encoding techniques are fully understood by users.

9: Ensure that the UI's appearance is *uncluttered*.

- Avoid displaying too much information.
- Organize the information effectively.

Usability Principles

10: Consider the needs of *different groups* of users.

- Accommodate people from different *locales* and people with *disabilities*.
- Ensure that the system is usable by both *beginners* and *experts*.

11: Provide all necessary *help*.

- Organize help well.
- Integrate help with the application.
- Ensure that the help is accurate.

Usability Principles

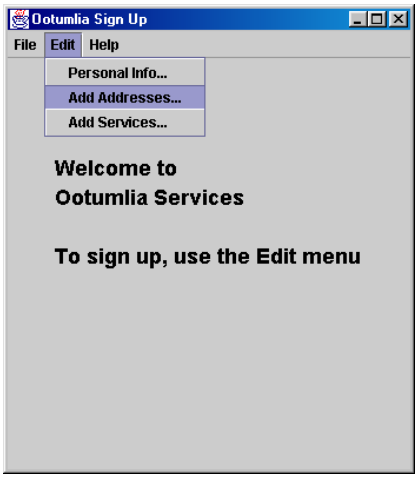
12. Be *consistent*.

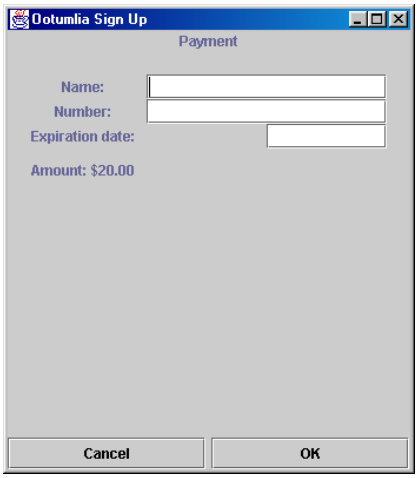
- Use similar layouts and graphic designs throughout your application.
- Follow look-and-feel standards.
- Consider mimicking other applications.

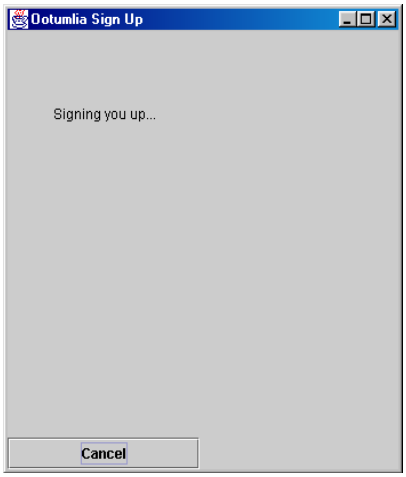
Some encoding techniques

- Text and fonts
- Icons
- Photographs
- Diagrams and abstract graphics
- Colors
- Grouping and bordering
- Spoken words
- Music
- Other sounds
- Animations and video
- Flashing

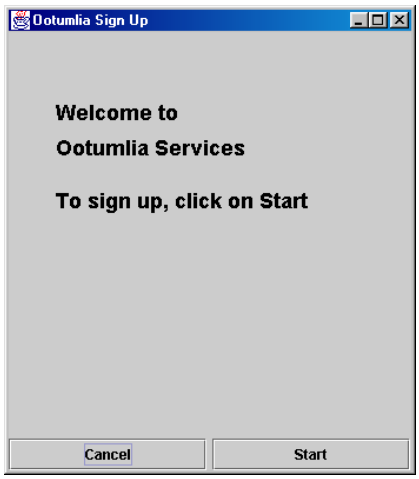
Example
(bad UI)

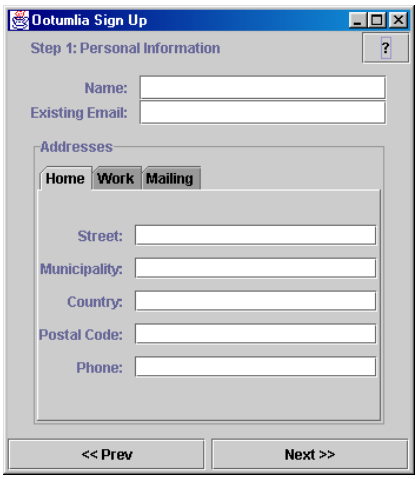


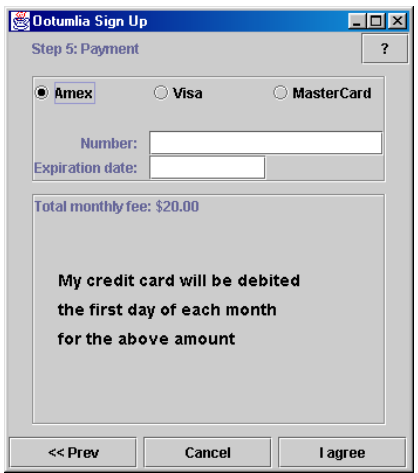


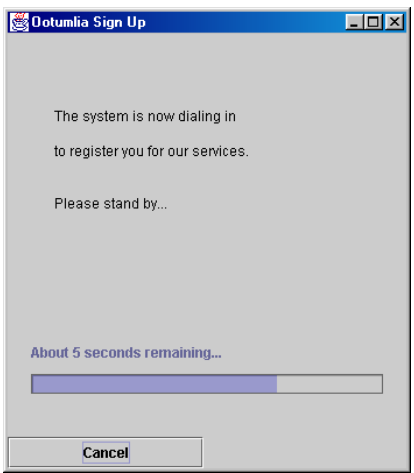


Example
(better UI)









7.5 Evaluating User Interfaces

Heuristic evaluation

1. Pick some use cases to evaluate.

2. For each window, page or dialog that appears during the execution of the use case
   — Study it in detail to look for possible usability defects.

3. When you discover a usability defect write down the following information:
   — A short description of the defect.
   — Your ideas for how the defect might be fixed.

Evaluating User Interfaces

Evaluation by observation of users

- Select users corresponding to each of the most important actors
- Select the most important use cases
- Write sufficient instructions about each of the scenarios
- Arrange evaluation sessions with users
- Explain the purpose of the evaluation
- Preferably videotape each session
- Converse with the users as they are performing the tasks
- When the users finish all the tasks, de-brief them
- Take note of any difficulties experienced by the users
- Formulate recommended changes

7.6 Implementing a Simple GUI in Java

The Abstract Window Toolkit (AWT)

- **Component**: the basic building blocks of any graphical interface.
  — Button, TextField, List, Label, ScrollBar.
- **Container**: contain the components constituting the GUI
  — Frame, Dialog and Panel
- **LayoutManager**: define the way components are laid out in a container.
  — GridLayout, BorderLayout

Example


```
public class ClientGUI extends Frame implements ChatIF
{
  private Button closeB = new Button("Close");
  private Button openB = new Button("Open");
  private Button sendB = new Button("Send");
  private Button quitB = new Button("Quit");
  private TextField portTxF = new TextField("12345");
  private TextField hostTxF = new TextField("localhost");
  private TextField message = new TextField();
  private Label portLB = new Label("Port: ", Label.RIGHT);
  private Label hostLB = new Label("Host: ", Label.RIGHT);
  private Label messageLB = new Label("Message: ",
Label.RIGHT);
  private List messageList = new List();
 ...
}
```

Example

```
public ClientGUI(String host, int port)
{
   super("Simple Chat");
   setSize(300,400);
   setVisible(true);
   setLayout(new BorderLayout(5,5));
   Panel bottom = new Panel();
   add("Center", messageList);
   add("South", bottom);
   bottom.setLayout(new GridLayout(5,2,5,5))
   bottom.add(hostLB);
   bottom.add(hostTxF);
   bottom.add(portLB);
   bottom.add(portTxF);
   bottom.add(messageLB);
   bottom.add(message);
   bottom.add(openB);
   bottom.add(sendB);
   bottom.add(closeB);
   bottom.add(quitB);
   ...
}
```


Example

```
sendB.addActionListener(new ActionListener()
{
       public void actionPerformed(ActionEvent e)
       {
              send();
       }
});


public void send()
{
   try
  {
      client.sendToServer(message.getText());
   }
   catch (Exception ex)
  {
      messageList.add(ex.toString());
      messageList.makeVisible(messageList.getItemCount()-1);
      messageList.setBackground(Color.yellow);
   }
}
```

7.7 Difficulties and Risks in UI Design

- **Users differ widely**
  - —*Account for differences among users when you design the system.*
  - —*Design it for internationalization.*
  - —*When you perform usability studies, try the system with many different types of users.*
- **User interface implementation technology changes rapidly**
  - —*Stick to simpler UI frameworks widely used by others.*
  - —*Avoid fancy and unusual UI designs involving specialized controls that will be hard to change.*

Difficulties and Risks in UI Design

- **User interface design and implementation can often take the majority of work in an application**:
  - —*Make UI design an integral part of the software engineering process.*
  - —*Allocate time for many iterations of prototyping and evaluation.*
- **Developers often underestimate the weaknesses of a GUI**
  - —*Ensure all software engineers have training in UI development.*
  - —*Always test with users.*
  - —*Study the UIs of other software.*