User interface design

Lecture 22

Agenda

- understand a number of user interface design principles;
- several interaction styles and appropriateness;
- understand when to use graphical and textual presentation of information;
- know what is involved in the principal activities in the use interface design process;
- understand usability attributes and have been introduced to different approaches to interface evaluation.

Design issues

- How should the user interact with the computer system?
- How should information from the computer system be presented to the user?

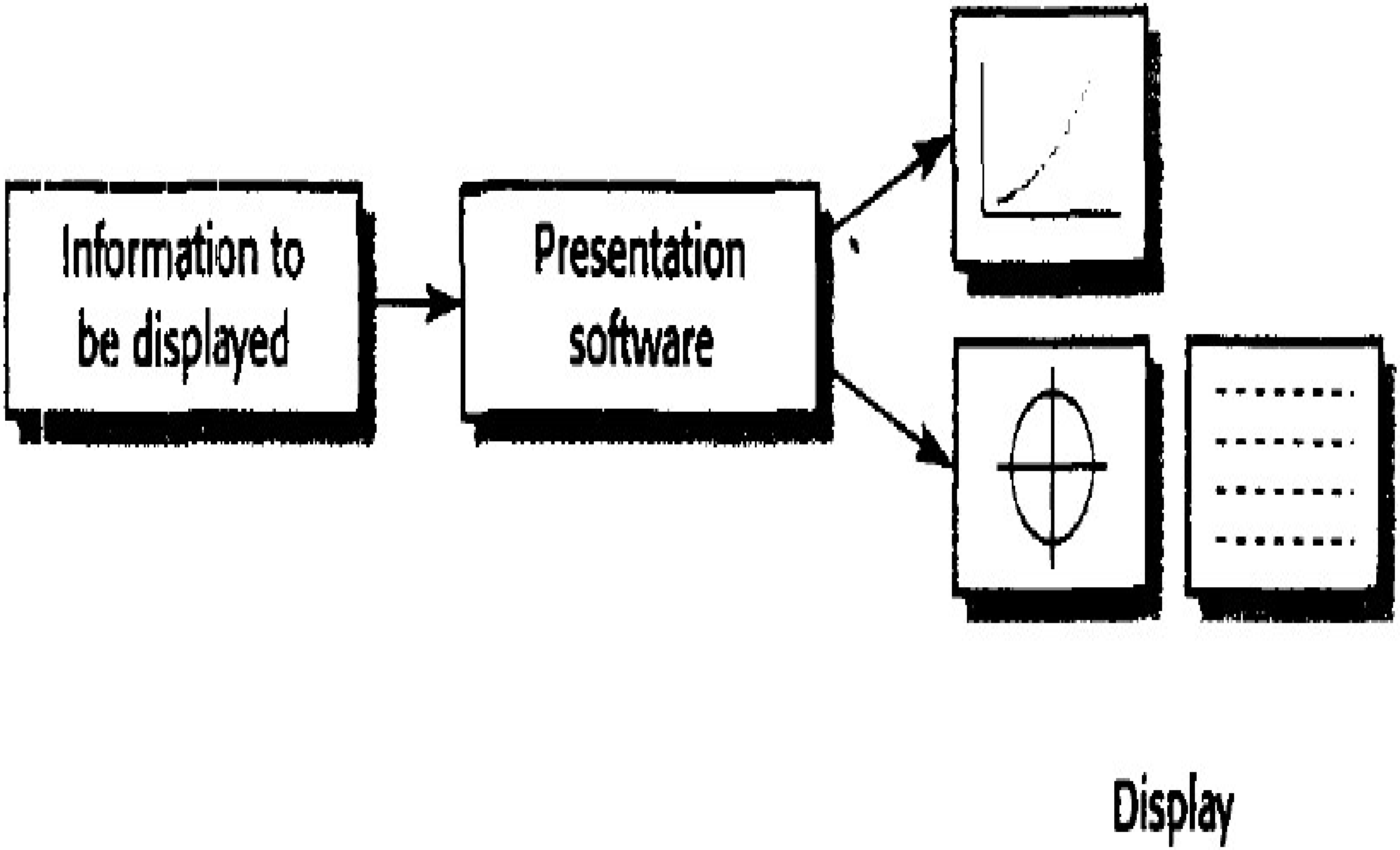
User interaction styles

- Direct *manipulation*
- *Menu selection*
- *Form fill in*
- *Command language*
- *Natural language*

Interaction style	Main advantages	Main disadvantages	Application examples
Direct manipulation	Fast and intuitive interaction Easy to learn	May be hard to implement Only suitable where there is a visual metaphor for tasks and objects	Video games CAD systems
Menu selection	Avoids user error Little typing required	Slow for experienced users Can become complex if many menu options	Most general-purpose systems
Form fill-in Easy to learn Checkable	Simple data entry	Takes up a lot of screen space Causes problems where user options do not match the form fields	Stock control Personal loan processing
Command language	Powerful and flexible	Hard to learn Poor error management	Operating systems Command and control systems
Natural language	Accessible to casual users Easily extended	Requires more typing Natural language understanding systems are unreliable	Information retrieval systems

Information presentation

- All interactive systems have to provide some way of presenting information to users. The information presentation may simply be a direct representation of the input information (e.g., text in a word processor) or it may present the information graphically.
- A good design guideline has to keep the software required for information presentation separate from the information itself.
- Separating the presentation system from the data allows us to change the representation on the user s screen without having to change the underlying computational system.



Information to be displayed

Presentation software

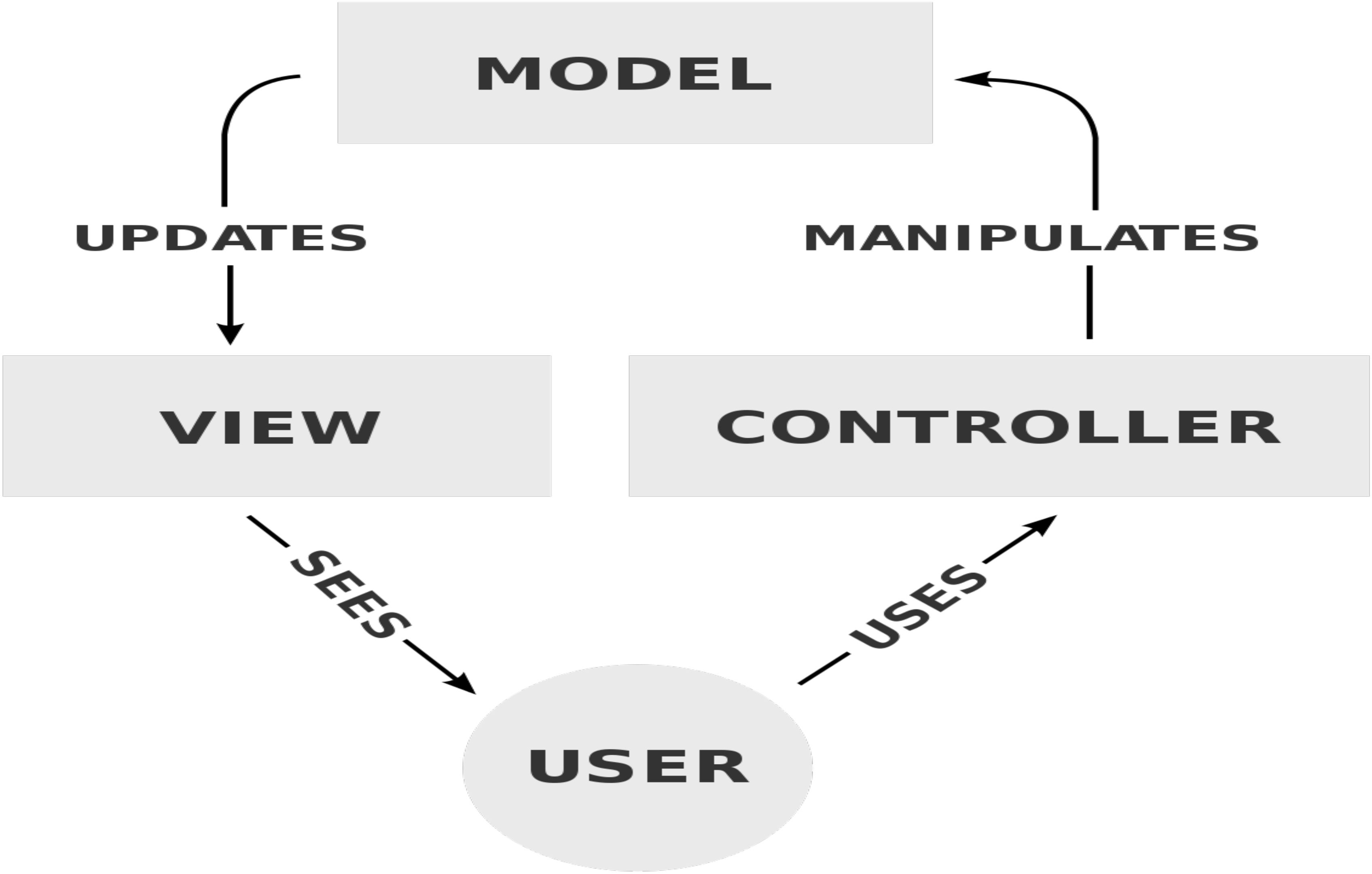
Display

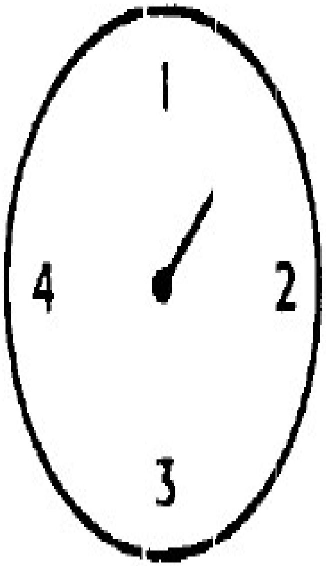
MVC Approach

- The MVC approach (Figure 16.6), first made widely available in Smalltalk (Goldberg and Robson, 1983), is an effective way to support multiple presentations of data.
- Users can interact with each presentation in a style that is appropriate to the presentation. The data to be displayed is encapsulated in a model object.
- Each model object may have a number of separate view objects associated with it where each view is a different display representation of the model.

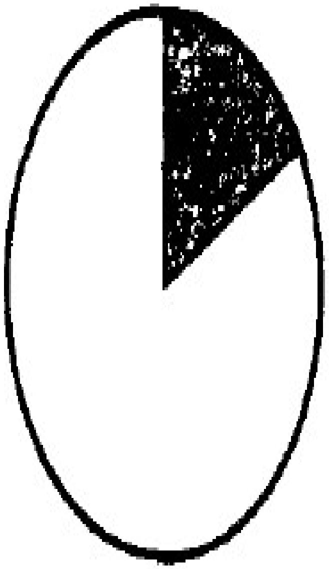
MVC Approach

- Each view has an associated controller object that handles user input and device interaction. Therefore, a model that represents numeric data may have a view that represents the data as a histogram and a view that presents the data as a table.
- The model may be edited by changing the values in the table or by lengthening or shortening the bars in the histogram.

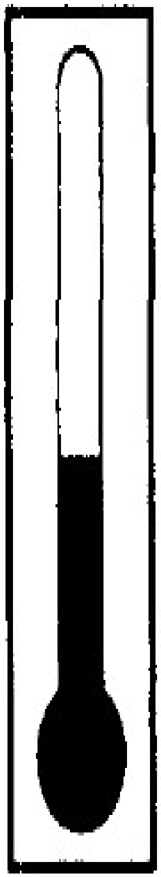

MODEL
UPDATES
MANIPULATES
VIEW
CONTROLLER
SEES
USES
USER



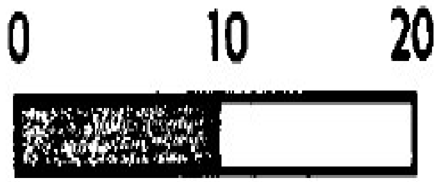
Dial with needle



Pie chart



Thermometer



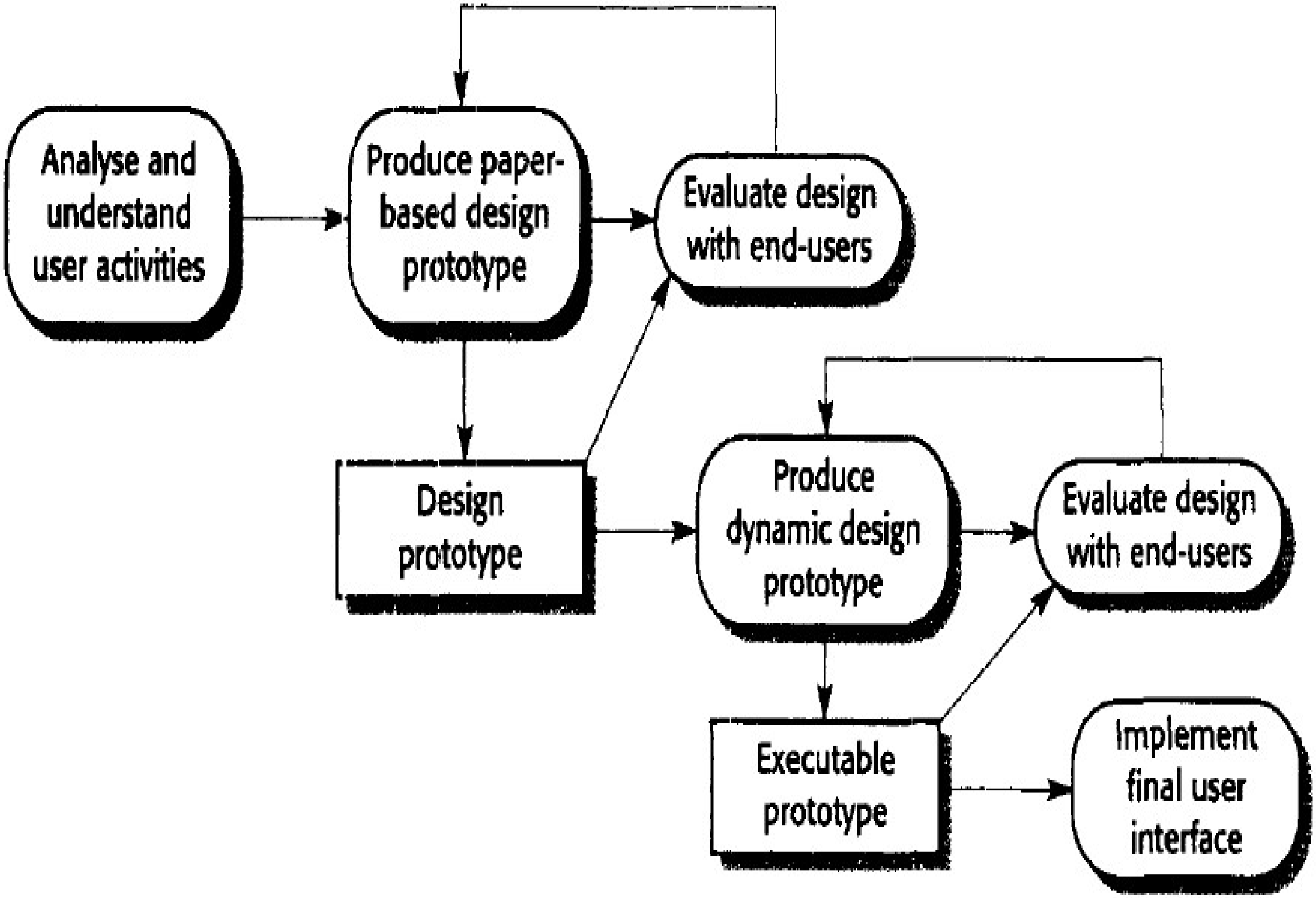
Horizontal bar

Factor	Description
Context	Wherever possible, the messages generated by the system should reflect the current user context. As far as is possible, the system should be aware of what the user is doing and should generate messages that are relevant to their current activity
Experience	As users become familiar with a system they become irritated by long, 'meaningful' messages. However, beginners find it difficult to understand short, terse statements of a problem. You should provide both types of message and allow the user to control message conciseness.
Skill level	Messages should be tailored to the users' skills as well as their experience. Messages for the different classes of users may be expressed in different ways depending on the terminology that is familiar to the reader.
Style	Messages should be positive rather than negative. They should use the active rather than the passive mode of address. They should never be insulting or try to be funny.
Culture	Wherever possible, the designer of messages should be familiar with the culture of the country where the system is sold. There are distinct cultural differences between Europe, Asia and America. A suitable message for one culture might be unacceptable in another.

The UI design process

- User interface (UI) design is an iterative process where users interact with designers and interface prototypes to decide on the features, organization and the look and feel of the system user interface.
- Sometimes, the interface is separately prototyped in parallel with other software engineering activities.
- More commonly, especially where iterative development is used, the user interface design proceeds incrementally as the software is developed.
- In both cases, however, before you start programming, you should have developed and, ideally, tested some paper-based designs.

Design Process


Analyse and understand user activities
Produce paper-based design prototype
Evaluate design with end-users
Design prototype
Produce dynamic design prototype
Evaluate design with end-users
Executable prototype
Implement final user interface

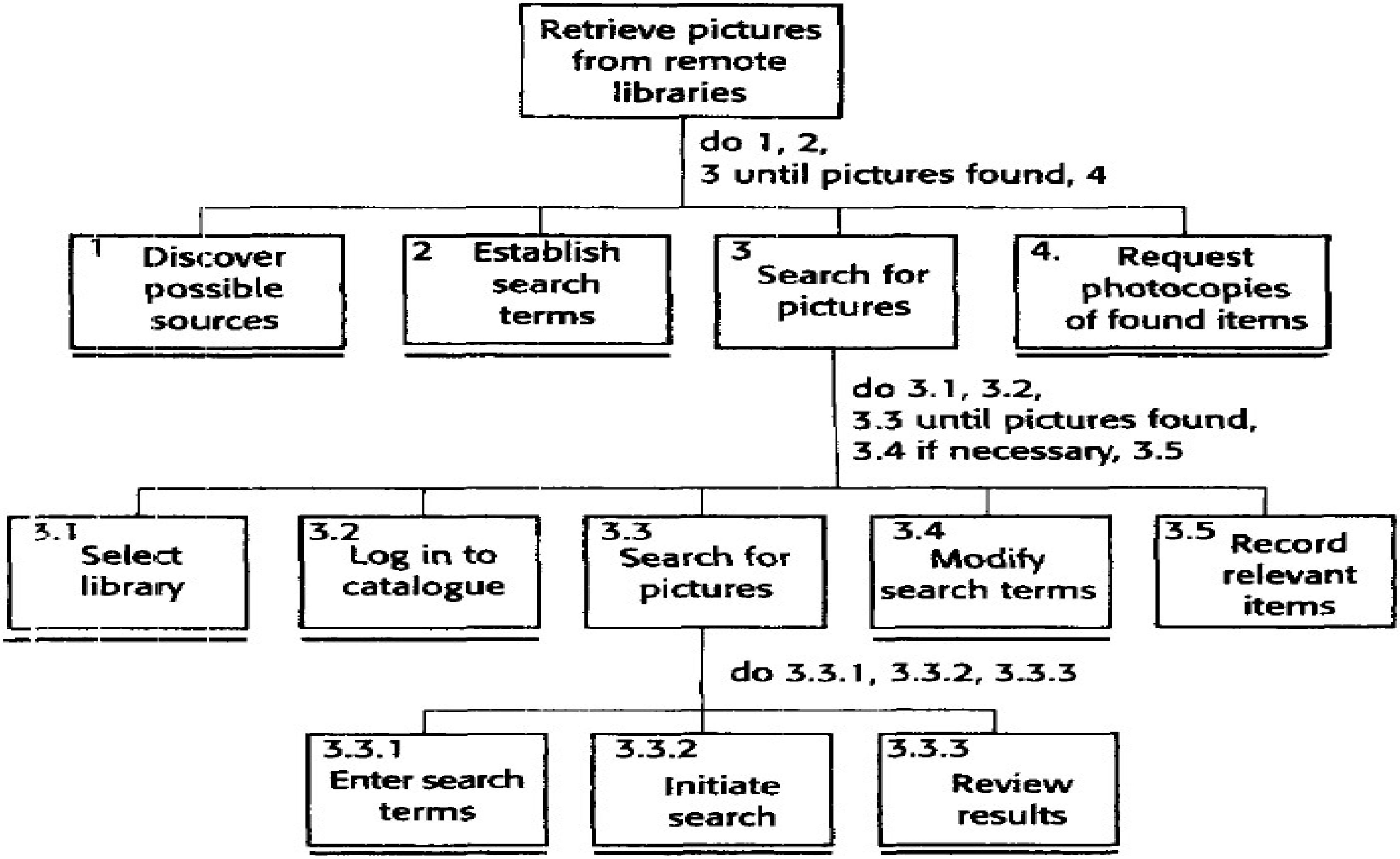
User analysis

- If you don't understand what users want to do with a system, then you have no realistic prospect of designing an effective user interface.
- To develop this understanding, you may use techniques such as task analysis, ethnographic studies, user interviews and observations or, commonly, a mixture of all of these.

User analysis problems

- Notations such as UML sequence charts may be able to describe user interactions and are ideal for communicating with software engineers.
- However, other users may think of these charts as too technical and will not try to understand them.
- It is very important to engage users in the design process, you therefore usually have to develop natural language scenarios to describe user activities.

LIBSYS system Natural Language scenario



User interface prototyping

- Evolutionary or exploratory prototyping with end-user involvement is the only practical way to design and develop graphical user interfaces for software systems.
- Involving the user in the design and development process is an essential aspect of user-centred design,a design philosophy for interactive systems.

Prototyping process

- adopt a two-stage prototyping process:
  - develop paper prototypes-mock-ups of screen designs-and walk through these with end-users.
  - develop increasingly sophisticated automated prototypes, then make them available to users for testing and activity simulation.

Approaches -user interface prototyping:

- There are three approaches that you can use for user interface prototyping:
  - – *Script-driven approach*
  - – *Visual programming languages*
  - – *Internet based prototyping*

Script-driven approach

- In this approach, you create screens with visual elements, such as buttons and menus, and associate a script with these elements.
- When the user interacts with these screens, the script is executed and the next screen is presented, showing them the results of their actions. There is no application logic involved.

Visual programming languages

- Visual programming languages, such as Visual Basic, incorporate a powerful development environment, access to a range of reusable objects and a user-interface development system that allows interfaces to be created quickly, with components and scripts associated with interface objects.

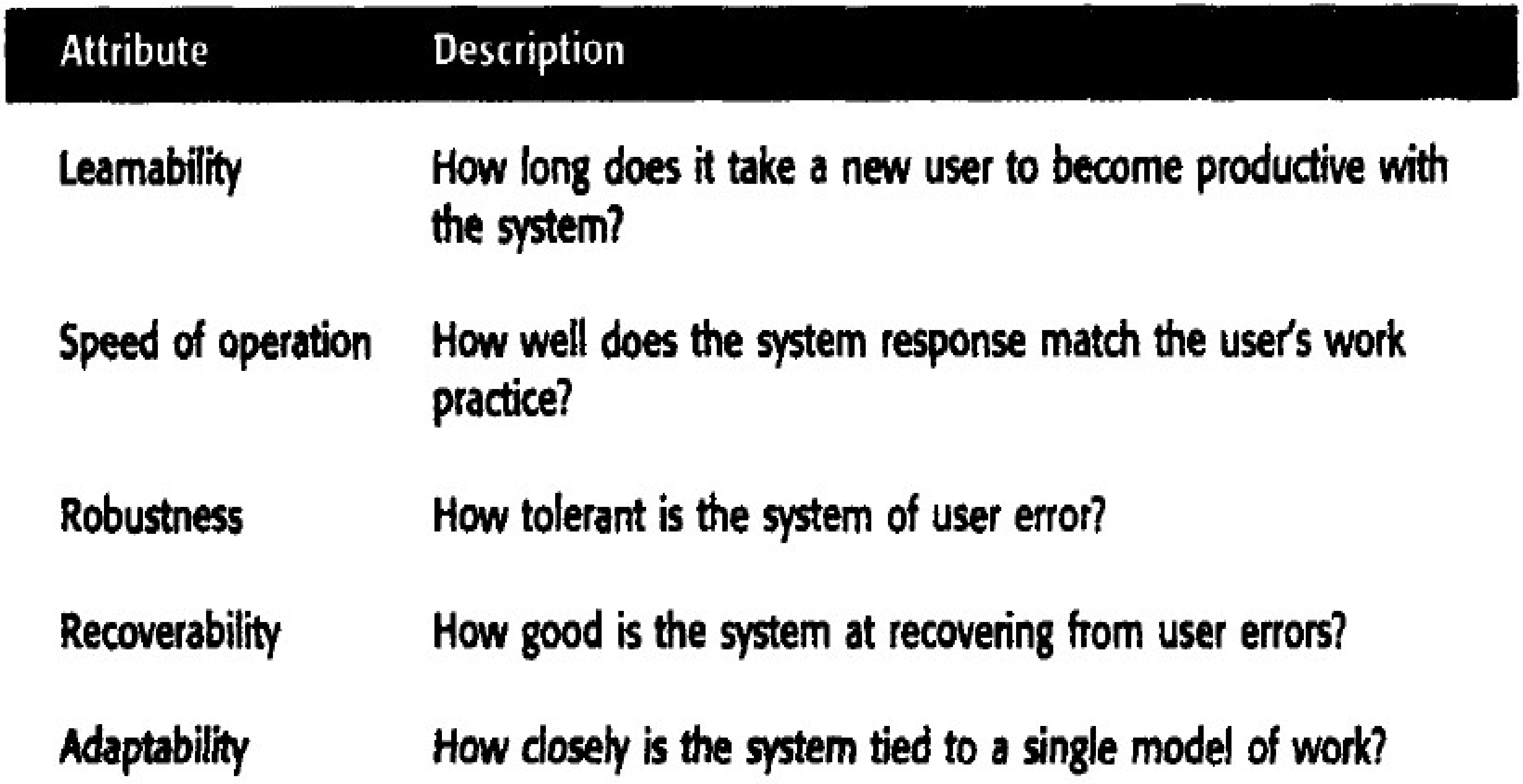
Internet based prototyping

- These solutions, based on web browsers and languages such as Java, offer a ready-made user interface
- add functionality by associatmg segments of Java programs with the information

Interface evaluation

- Interface evaluation is the process of assessing the usability of an interface and checking that meets user requirements. Therefore, it should be part of the normal verification and validation process for software systems

Usability attributes



Attribute	Description
Learnability	How long does it take a new user to become productive with the system?
Speed of operation	How well does the system response match the user's work practice?
Robustness	How tolerant is the system of user error?
Recoverability	How good is the system at recovering from user errors?
Adaptability	How closely is the system tied to a single model of work?

user interface evaluation

- Questionnaires that collect information about what users thought of the interface;
- Observation of users at work with the system and 'thinking aloud' about how they are trying to use the system to accomplish some task;
- Video 'snapshots' of typical system use;
- The inclusion in the software of code which collects information about the most used facilities and the most common errors.

Reference

- Chapter 16; Software Engineering
- Author Ian Sommerville