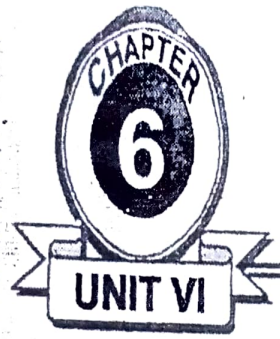

Task Models and Dialogs

Syllabus

Task Analysis, DOET (Design of Everyday Things). Design Dialogs Notations, Warnings, and Error messages. Model-based Evaluation. User Testing, Usability Testing, User Acceptance Testing.

6.1 Introduction

Objective : To implement simple graphical user interfaces based on principles of HCI.

Outcome : Apply the fundamental aspects of designing and evaluating interfaces.

Apply appropriate HCI techniques to design systems that are usable by people.

Important Discussions :

- Task models and dialogs are important as the last phase involved in HCI system. User models based on various HCI systems acts as input to task models. In this context, appropriate use of task models along with dialogs for interaction are required. DOET (Design of Everyday Things) is necessary for presenting design thing on paper along with prototyping.
- Additionally, design dialog notations are used to provide help, warning and error messages to users. Also, testing is essential to test Implementation phase that deals with coding using different of techniques. Testing of HCI system is categorized as user testing, usability testing, and user acceptance testing.

6.2 Task Analysis

Task analysis provides the study of performing the tasks and comparative performance analysis of existing HCI systems. Tasks may be divided into multiple sub tasks using modularity. Multiple tasks may work as the part of parallel processing. There are three techniques used in task analysis namely;

1. Tasks are divided into subtasks
2. Tasks are classified into various categories
3. Tasks are listed

- Task analysis is used to provide guideline about sequence of tasks to be performed by the user selection appropriate methodology used solve tasks in a correct way.
- It is effectively used in deciding about action path of HCI process.
- E.g. Project is divided into 5 modules and is distributed in to 5 users then the time require complete task is reduces in the multiples of time and customer gets more outcome within time.
- There exists three approaches for analysing the tasks as follows:
  1. Task decomposition
  2. Knowledge-based techniques
  3. Entity-relationship techniques
- Task decomposition is used to divide main task into multiple sub tasks.
- Knowledge-based techniques dependant on object and operations exists in task. Here, raw da transformed into information, and information is converted into knowledge after operations.
- Entity-relation-based technique deals with object oriented approach and interaction bet various objects and entities.
- Moreover, computer based tasks are the parts of computer system. Therefore, word proce power point presentation, Excel applications acts as various tasks.
- Multitasking is related to parallel processing in HCI System.
- Task analysis is different from traditional analysis of HCI System.
- The major role of task analysis is user-based, outcome-based, and performance-based approa

6.3 DOET (Design of Everyday Things)

Overviews

Why are some everyday things difficult to understand and use?

Psychopathology

- We are surrounded by many everyday things that have poor usability.
  (i) Programming a VCR

Telephone features we can't remember how to use.

(iii) Photocopiers and fax machines face down or up ?

Many of these things can be difficult to interpret and frustrating to use if they provide no clues or false clues as to how they operate.

Why usability is important ?

Poor usability results in

- Anger and frustration
- Decreased productivity in workplace
- Higher error rates
- Physical and emotional injury
- Equipment damage
- Loss of customer loyalty
- Costs money

Usability is a measure of the effectiveness, efficiency and satisfaction with which specified uses can achieve specified goals in a particular environment.

Example of Poor Design

Wireless PowerPoint slide controller

Short press to go forward.

Long press to go backward.

efrigerator temperature control

Two compartments and two controls

One cooling unit.

orman's Principles of Design

Make things visible.

Provide good conceptual design.

(i) Affordance

(ii) Mapping

ii) Constraints

Feedback

Norman's Principles in software

1. **Visibility**
   - Visibility of the tasks the interface supports.
   - Communication of system state/mode.
2. **Affordance**

   If it looks like a button it can be pressed, if it is a underlined it can be clicked (web).
3. **Mapping**

   Clicking on a particular interface element produces expected effect (under file should be open).
4. **Constraints**

   Constraining search criteria, graying out menu items that don't apply in a particular context.
5. **Feedback**

   Providing clear and immediate feedback for each user action.

6.4 Design Dialog Notations

6.4.1 What is Dialog ?

- Dialog, as opposed to a monolog, is a conversation between two or more parties. It has also come imply a level of cooperation or at least intent to resolve conflict.
- In the design of user interfaces, the dialog has a more specific meaning, namely the structure of the conversation between the user of the computer system.
- We can look at computer language at three levels
  - (i) Lexical
  - (ii) Syntactic
  - (iii) Semantic

Structured human dialogs

In contrast to most human conversant dialog with computers is relatively structured and constrained. It is only on star trek that one can freely chat to the computer and expect a response. So, where as in human conversation the grammar rules often stop once we get to the level of a sentence, those for computer dialogs may encompass the whole of the interaction.

6.4.2 Dialog Design Notations

- Dialog design notations used to ease of analysis and separation of the interface elements of the program from the actual calculations.

- For using a special notation is to write down the dialog before a program is written. This allows the designer to analyse the proposed structure, or perhaps use a prototyping tool to execute the dialog.
- A dialog notation is also a way for the members of a design team to talk about the design and eventually for the designer to pass on the intended dialog to the programmer of the actual application. Thus dialog notations often form an integral part of prototyping methodologies and tools.

6.4.3 Diagrammatic Notations

- Diagrammatic notations are heavily used in dialog design. At their best, they allow the designer to see at a glance the structure of the dialog. However, they often have trouble coping with more extensive or complex dialog structures.

6.4.3.1 State Transition Networks (STN)

- Consider a simple mouse. Based drawing tool. It has a menu with two options, circle of line and drawing surface. If you select circle you allowed to click on two further points on drawing circle surface. The first of these is the circle's centre of the second any point on the circumference. After the first point is selected, the system draws a rubber band line between the centre of the current mouse position. After the second point is chosen, the circle is drawn.
- The line option in the menu is to draw a poly line. That is, the user can select any number of points on the drawing surface which the system connects with straight lines. The last point is denoted by a double click on the mouse. Again system 'rubber bands' between successive mouse option state transition network is depicted in Fig. 6.4.1.

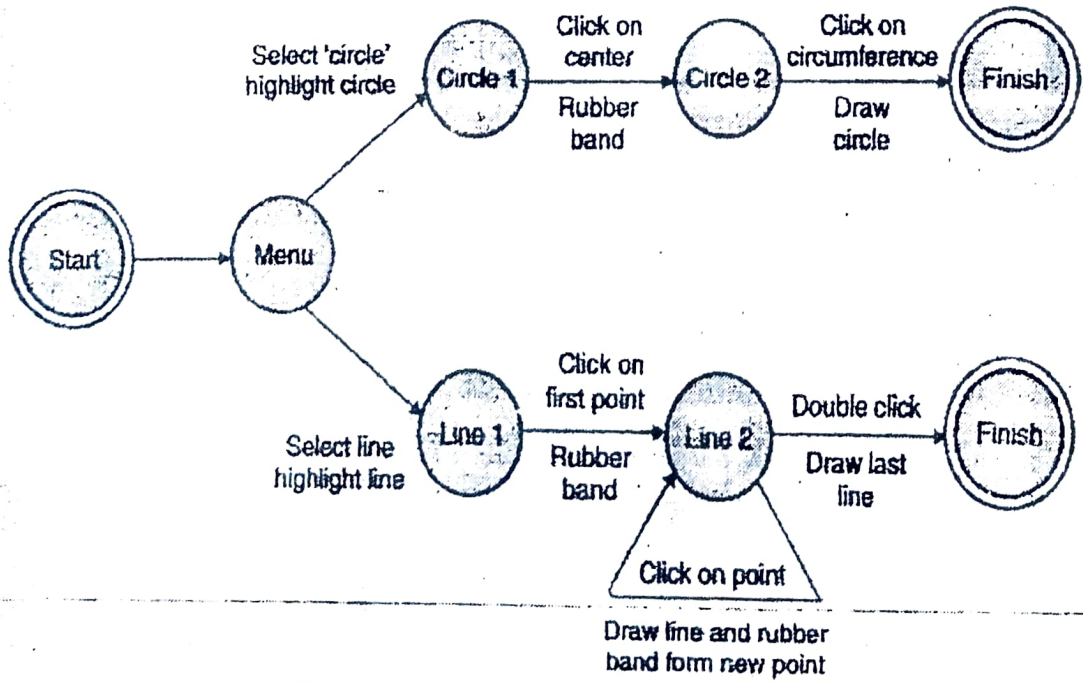

Fig. 6.4.1 : STN for Menu-driven Tools

6.4.3.2 Hierarchical State Transition Nets (HSTN)

The start of finish states are not real states, but are there merely to let us glue this bit of d into a bigger dialog.

Example

- The drawing tool may have a main menu, from which we can select one of these submenu graphics menu (circle, and lines), a tent menu (adding labels) and paint menu (free hand draw We would describe this complete system using the hierarchical STN as shown in Fig. 6.4.2.
- HSTN is like previous STN's but has additional composite states represented as rectangles w picture of a little STN in them. Each of these rectangle denotes the whole STN for the rele submenu.

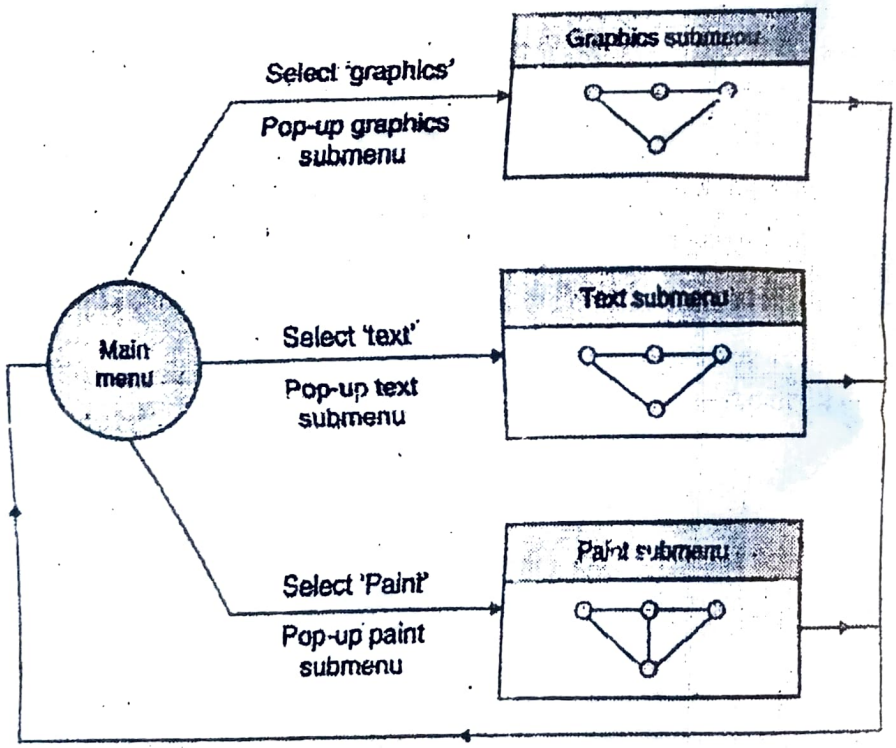

Fig. 6.4.2 : Hierarchical STN for complete drawing tool

6.4.3.3 Concurrent Dialogs and Combinatorial Explosion of States

Example :

A simple dialog box for describing tent style as one might find in a word processor. The contains three toggles, bold, italic and underline styles. A piece of tent can be emboldened, italici underlined or any combination of these three. To select say, emboldening, the user clicks over the toggle. To deselect it, the user simply click again. Simple dialog box with three toggles is represente Fig. 6.4.3 and Detailed state transition is shown in Fig. 6.4.4.

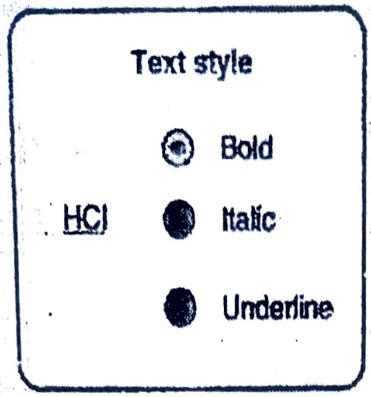

Fig. 6.4.3 : Simple Dialog Box with Three Toggles

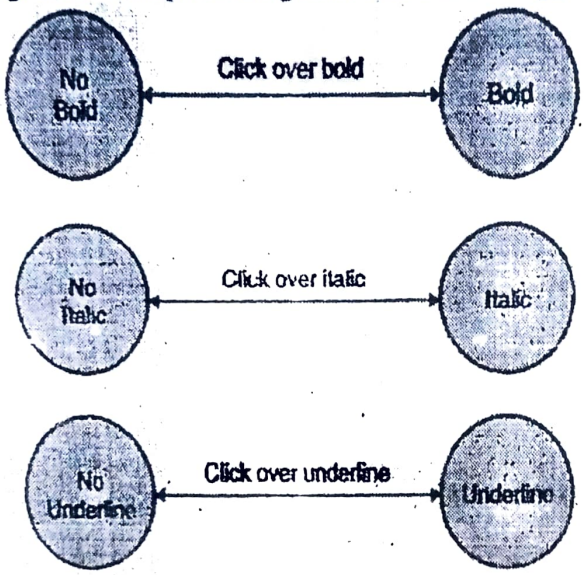

Fig. 6.4.4 : Individual Bold, Italic, Underline State Transition

6.4.3.4 Escapes and Help

This system pose problems that are similar to the combinatorial explosion from concurrent dialogs. Imagine that we have been observing the use of the drawing tool. We have noticed that users often find they have wrongly selected some option and want to get back to the menu. As the dialog is currently specified, once they select, say, the circle option, they must select two points before they are allowed to continue.

As a solution to this problem, we want to add an escape key, which, wherever you are, cancels what you are doing and returns you to the main menu. This seems quite a simple addition – it only took a sentence to say. However, to add it to the STN describing the system would require an arc from every state back to the main menu. STN for drawing tools with escapes is represented in Fig. 6.4.5.

Esc. Escape

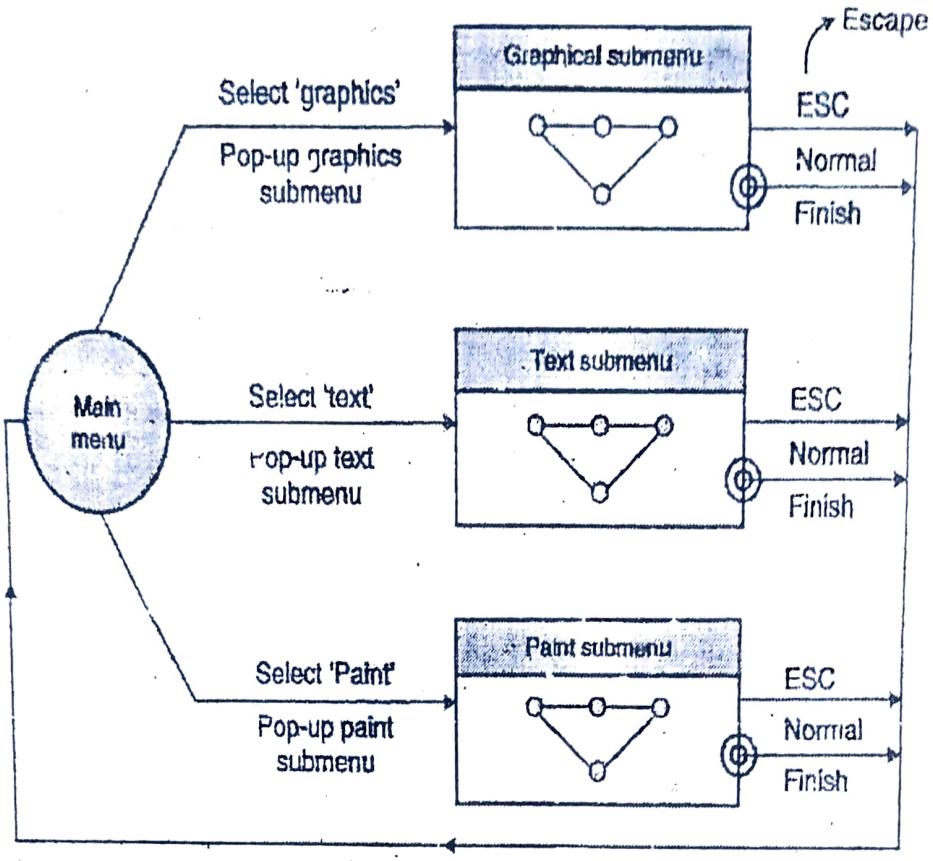

Fig. 6.4.5 : STN for Drawing Tools with Escapes

6.4.3.5 Petri Nets

- One of the oldest formalisms in computing science is the Petri net. It is a graphical formalism designed for reasoning about concurrent activities. In recent years it has been used by several researchers to specify aspects of single-user and multi user systems.
- In an STN the system is always at exactly one state. Indeed, you can simulate the behaviour of the system by moving a counter around the STN following arcs. A Petri net is similar except that the system has several 'states' at once. These are depicted as several black counters.
- The Fig. 6.4.6 shows a Petri net for a system having two near independent bold/italic toggles. The circles are called places (states) and the thin rectangles are called transitions.

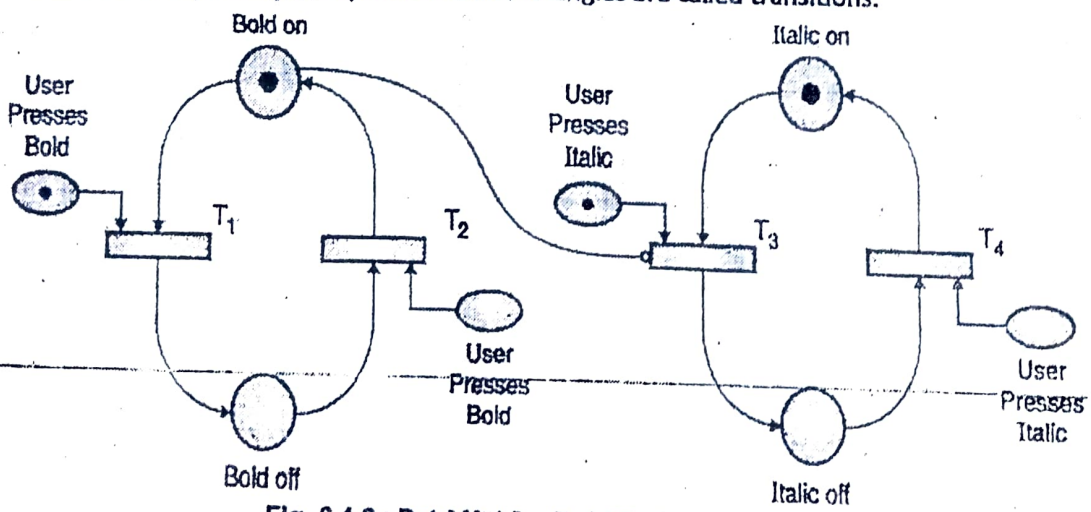

Fig. 6.4.6 : Petri Net for Bold /Italic Selection

6.4.3.6 State Charts

- Here is state chart can be seen as a form of STN. They were developed as a way of visually specifying complex reactive systems and address many of the problems, described earlier i.e. concurrent and escapes.
- The hierarchy in state charts is used within a single diagram to add structure, and to show which parts represent alternative states and which represent concurrent activity.

Example :

Fig. 6.4.7 is a state chart of a television control panel. The controller has five buttons labeled 'ON', 'OFF', 'MUTE', 'SEL' and 'RESET', the T.V. can either be ON or In standby mode. Imagine we start at standby and so on.

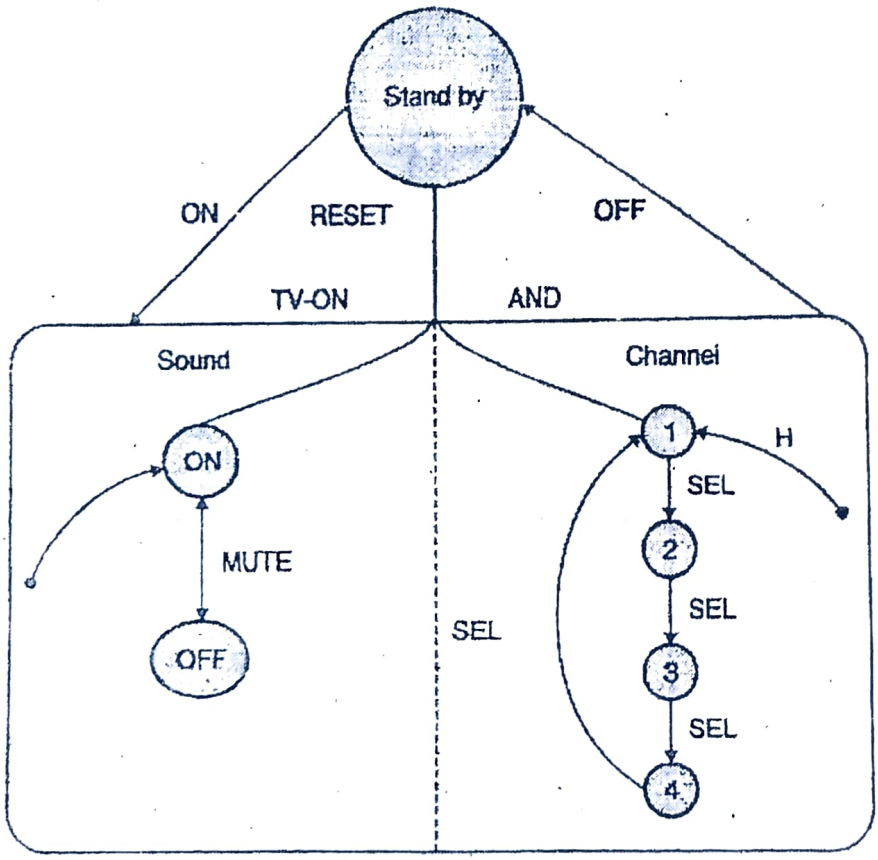

Fig. 6.4.7 : State chart for television control panel

6.4.3.7 Flow Charts

- Flow charts in various forms are perhaps the most widely used of any diagrammatic notation for programming. They can also be a simple, but useful, tool for dialog.
- In expressive power, they differ little from STN'S and share the problems of concurrency, escapes and so on.

- However, within the area of simple dialogs, they have the advantage of simplicity and the added benefit that most programmers will know what they mean.
- The boxes in flow chart represent processes or decisions and are thus not equivalent to the states of an STN.

Example :

Flowchart of Addition of two numbers is as shown in Fig. 6.4.8.

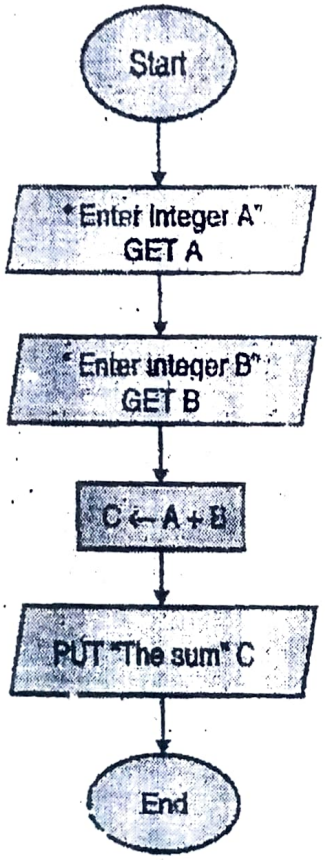

Fig. 6.4.8 : Flow chart

6.4.3.8 JSD Diagram

- JSD i.e. Jackson Structured Design. JSD, while not as old as flow charts, has been around for many years. During this time it has developed significantly, however it is one of the older parts of this methodology, the JSD diagram, which has been used for various aspects of task analysis and dialog system of design.
- As with flow charts, there may be an advantage to using JSD diagrams if they are already familiar to the programmers who will implement the dialog.
- Fig. 6.4.9 shows JSD diagram for top-level structure of an employee personnel system. The system allows the user to update the personnel record in various ways ; adding, displaying, altering, removing.

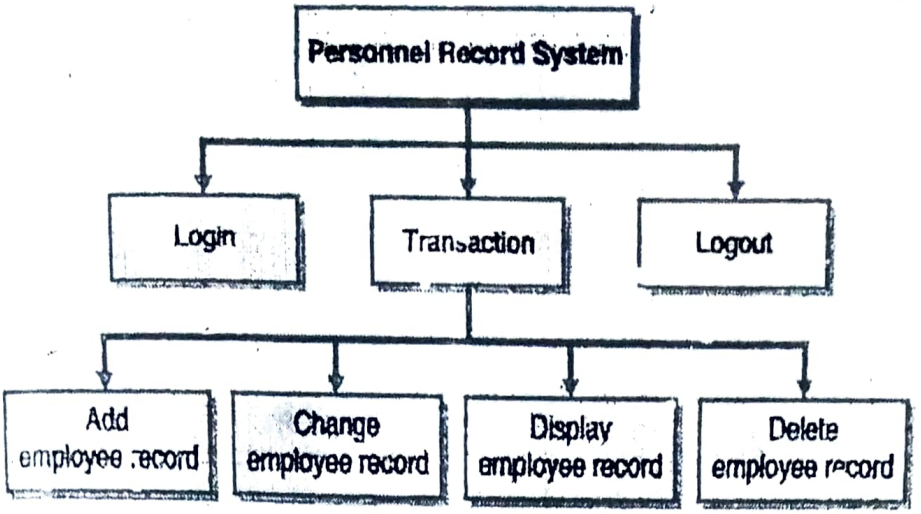

Fig. 6.4.9 : JSD Diagram for Personnel System

6.4 Textual Dialog Notations

Grammars

Production rules

Communicating sequential processes (CSP) and event algebras

Parameterized and dynamic interleaved dialog structure.

6.4.5 Dialog Semantic

- Notation specific semantic
- Links to programming languages
- Links to formal specification

6.5 Warning and Error Messages

Warning

Something has not worked as it should. This may be of greater or lesser importance depending on the circumstances.

Example

An input file was not found or was in wrong format.

Error

Something "serious" has gone wrong. This may require existing may be "recoverable" depending on the circumstances.

Example

The system has failed to allocate dynamic memory as requested.

6.6 Model based Evaluation

- Using cognitive architectures in user interface design is a form of usability evaluation models of the user.
- Approaches to developing usable system

6.6.1 Ensuring Usability : Standard Human Factors Process

- Early use of guidelines, empirical user testing of prototypes.
- Identify problems that impair learning or performance.
- Compare user performance to a specification.
- Standard human factors process is represented in Fig: 6.6.1.

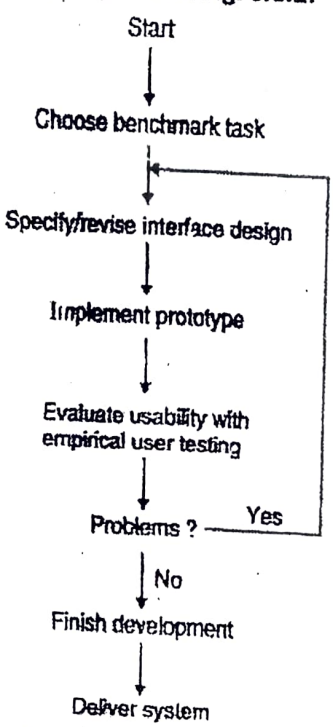

Fig. 6.6.1 : Standard human factors process

6.6.2 What's Good and Bad about Standard Human Factors Process

What's good ?

- Definitely works if used thoroughly enough.
- Much accumulated experience.

What's bad ?

- Slow and expensive.
- No systematic way to accumulate or analyze design experience.
- Only Psychology used.

6.6.3 Ensuring Usabiiity : Engineering Model Approach

- Use model predictions instead of user testing
  - (i) Describe the interface design in detail.
  - (ii) Build the model of user doing task.
  - (iii) Use the model to predict execution or learning time.
  - (iv) Revise or choose design depending on prediction.
- Get usability results before implementing prototype or user testing.
- Engineering model allows more design iterations
- Model summarizes the interface design from the user's point of view.
- Some user testing still required, be assured.

6.6.4 Engineering Model Process

- Use model first
- User testing for final check
- Loop back if necessary
- Engineering Model processes are shown in Fig. 6.6.2.

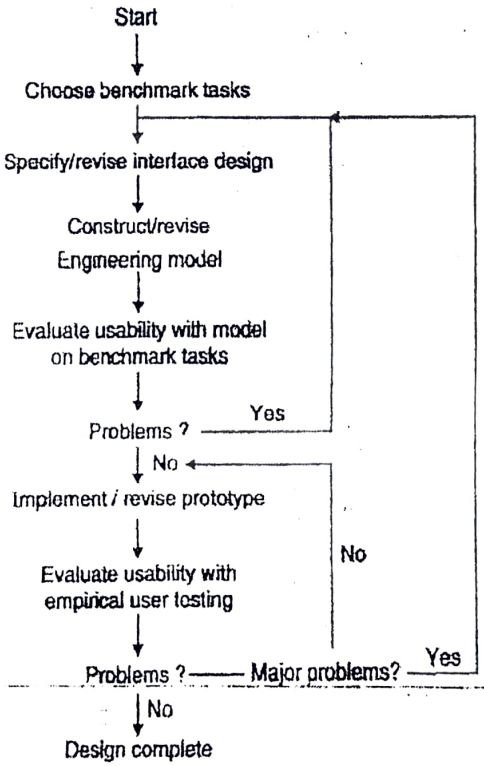

Fig. 6.6.2 : Engineering Model Process

6.6.5 Models in Science

- How good is the model as an account of what humans really do?
- Comparison of model to empirical data on human behaviour.
- Scientific standards on quality of the model, data and the comparison are quite demanding.
- Real world situation often impose practical barriers to scientific data collection, so standards are hard to meet outside the laboratory

6.6.6 Models in Design

- What does the model say about what people will do when they use the system for designing and development.
- Can't test model against data if the system is not built yet.
- Validating model against data is not part of the "normal" use of a model
- Model is known to be limited and approximate but can still be useful to design effort.

6.7 User Testing

- In HCI System phase-wise, module-wise and step-wise testing is required at each and every level. Because every level is dependent on correctness of previous level. User testing is important testing to evaluate about best suitable solution from multiple users based on their perspective and technical skill set.
- User testing is performed based on prediction analysis that includes historical knowledge, present knowledge and forecast knowledge about given problem. User testing provides outcome based approach at all the levels of HCI System.

6.8 Usability Testing

What is Usability Testing?

It is a non-functional testing technique that is a measure of how easily the system can be used by end users. It is difficult to evaluate and measure but can be evaluated based on the below parameters;

(1) Level of skill required to learn/ use the software. It should maintain the balance for both novice and expert user.

(2) Time required to get used to in using the software.

(3) The measure of increase in user productivity if any.

(4) Assessment of a user's attitude towards using the software.

is context, Usability Testing Process is shown in Fig. 6.8.1.

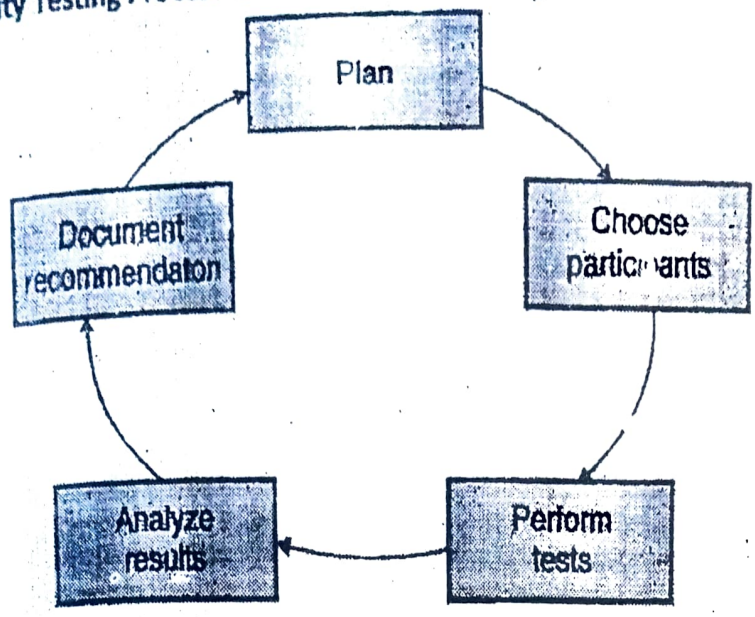

Fig. 6.8.1 : Usability Testing Process

6.9 User Acceptance Testing

What is User Acceptance Testing ?

- A testing methodology where the clients/ and users involved in testing the product is validate the product against their requirement. It is performed at client location at developers site.
- For industry such as medicine or aviation industry, contract and regulatory compliance testing and operational acceptance testing is also carried out as part of user acceptance testing.
- *UAT is content dependent* and the UAT plans are prepared based on the requirements and NOT *Mandatory to* execute all kinds of user acceptance tests and even coordinated and contributed by *testing team*. User accepting Testing is as shown in Fig. 6.9.1.

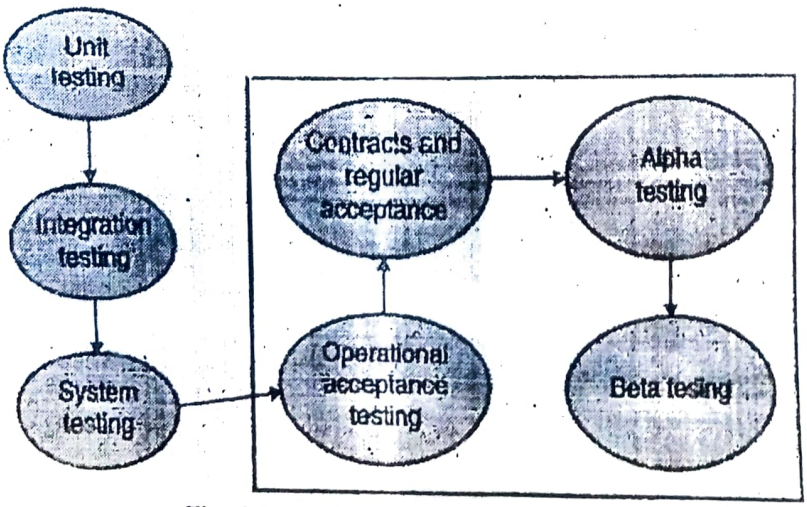

Fig. 6.9.1 : User Acceptance Testing

The acceptance test cases are executed against the test data or using an acceptance test script and then the results are compared with the expected ones.

Acceptance criteria

- Function correctness and completeness
- Data integrity
- Data conversion
- Usability
- Performance
- Timelines
- Confidentiality and availability
- Installability and upgradability
- Scalability
- Documentation

Acceptance Test Plan Attributes

- Introduction
- Acceptance test category
- Operation environment
- Test case ID
- Test objective
- Test procedure
- Test schedule
- Resources

Acceptance Test Report Attributes

- Report identifier
- Summary of Results
- Variations
- Recommendations
- Summary of To-Do list
- Approval decision

Review Questions

Q. 1 Explain task analysis, design dialogs notations, warning, and error messages in detail.

Q. 2 What is importance of Design of Every Day Things (DOET) in HCI?

Q. 3 What are the various phases used in Model-based Evaluation?

Q. 4 Write short note on following testing :

(a) User Testing

(b) Usability Testing

(c) User Acceptance Testing

Q. 5 What is role of the Hierarchical model representation in designing task and structure?

Q. 6 Explain Linguistic model used for the user-system grammar.

Q. 7 How to improve human motor skills using physical and device models?

Q. 8 Design cognitive architecture for HCI system using Hierarchical model, Linguistic model, physical and device models

Q. 9 Elaborate various techniques used for task analysis such as decomposition of tasks into subtasks, Taxonomic Classification of task knowledge, Listing things used and actions performed.

Q. 10 Explain sources of information based on existing documentation, observation, and interviews.

Q. 11 Elaborate manual and documentation ways for designing new systems using task analysis.

Q. 12 Write short note on:

(a) Task Decomposition

(b) Knowledge-based Techniques

(c) Entity-Relation-based Analysis

□□□