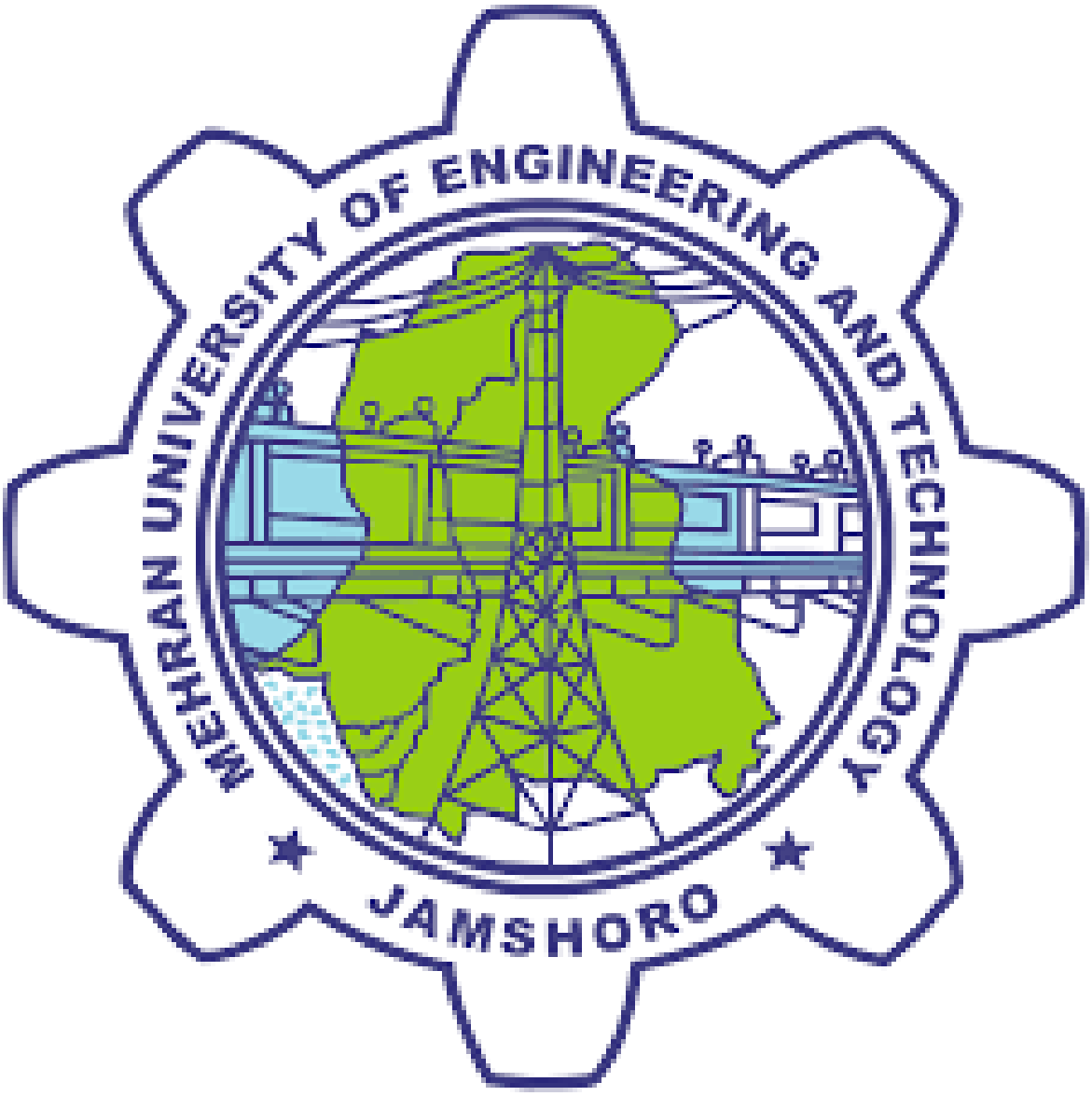
Name: ZOHAIB HASSAN SOOMRO

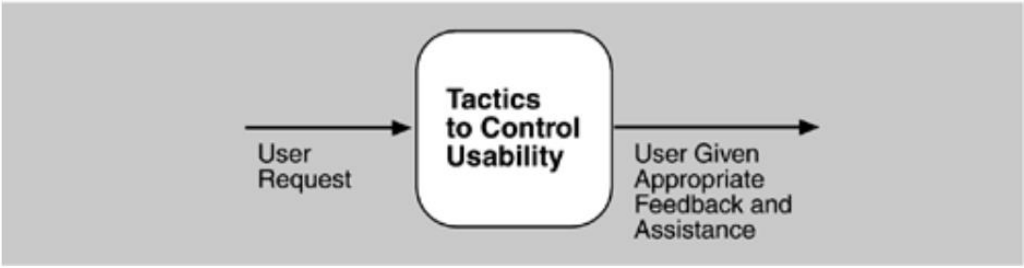
RollNo#: 19SW42

Subject: SDA



Usability Tactics:

Usability is concerned with how easy it is for the user to accomplish a desired task and the kind of support the system provides to the user. Two types of tactics support usability, each intended for two categories of "users." The first category, runtime, includes those that support the user during system execution. The second category is based on the iterative nature of user interface design and supports the interface developer at design time.



1. Run-Time Tactics:-

It is the management of applications at run-time. It includes:

Maintain a Model of the Task:

- It is to manage work patterns and their use by categories. Clearly manage each type of usage. Reduce duplication and have standards for users to understand and use, for example, a banking transaction system that has a model of use separated by location and device, separated by usage of branch staff and customers, separated by device into Operate via ATM, mobile phone or tablet, PC or notebook.

Maintain a Model of the User:

- It is to manage work and usage patterns by user type, such as screens for employees, separate screens for customers, displaying and working according to individual user rights that may be different. This technique can be very useful if users have a variety of characteristics or behaviours. Therefore, it is best to start by analysing and classifying user attributes or behaviours.

Maintain a Model of the System:

- It is the management of work patterns and usage by system. It determines the expected system behaviour so that appropriate feedback can be given to the user. The system model predicts items such as the time needed to complete current activity.
- For Example: Processing bank transactions that receive requests from ATMs should take a short time. Because ATMs are physical, if there are many people in line, it will make people wait longer.

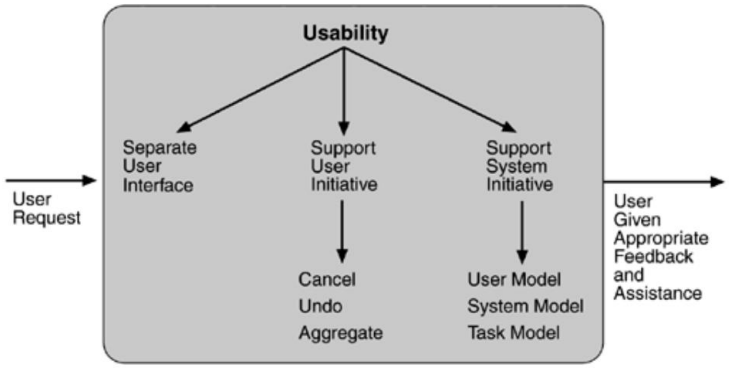
2. Design-Time Tactics:-

It is the management of applications at design-time, It includes:

Separate User Interface:

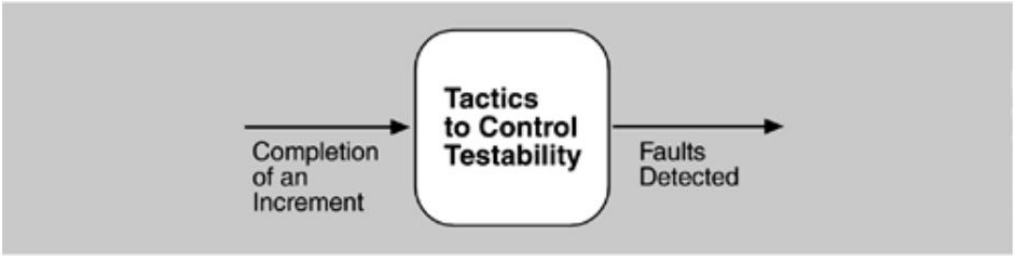
- It is the separation of the display or interface between the user and the functional part of the system, for example, the presentation of the application is separated from the application to allow for more independent management of the display, which may use different patterns. It helps a lot such as
    - Model-View-Controller (MVC)
    - Presentation-Abstraction-Control (PAC)

- View Helper
- Dispatcher View
- etc.



Testability Tactics:

The goal of tactics for testability is to allow for easier testing when an increment of software development is completed. Architectural techniques for enhancing the software testability have not received as much attention as more mature fields such as modifiability, performance, and availability, but, since testing consumes such a high percentage of system development cost, anything the architect can do to reduce this cost will yield a significant benefit.



The goal of a testing is to discover faults. This requires that input be provided to the software being tested and that the output be captured.

Executing the test procedures requires some software to provide input to the software being tested and to capture the output. This is called a test harness. A question that is not considered here is the design and generation of the test harness. In some systems, this takes substantial time and expense.

Two categories of tactics for testing: providing input and capturing output, and internal monitoring.

1. INPUT/OUTPUT:

   There are three tactics for managing input and output for testing.

- Record/Playback:

  It refers to both capturing information crossing an interface and using it as input into the test harness. The information crossing an interface during normal operation is saved in some repository and represents output from one component and input to another. Recording this information allows test input for one of the components to be generated and test output for later comparison to be saved.

- Separate interface from implementation:

  It allows substitution of implementations for various testing purposes. Stubbing implementations allows the remainder of the system to be tested in the absence of the component being stubbed. Substituting a specialized component allows the component being replaced to act as a test harness for the remainder of the system.

- Specialize access routes/interfaces:

  Having specialized testing interfaces allows the capturing or specification of variable values for a component through a test harness as well as independently from its normal execution. For example, metadata might be made available through a specialized interface that a test harness would use to drive its activities. Specialized access routes and interfaces should be kept separate from the access routes and interfaces for required functionality. Having a hierarchy of test interfaces in the architecture means that test cases can be applied at any level in the architecture and that the testing functionality is in place to observe the response.

2. INTERNAL MONITORING:

A component can implement tactics based on internal state to support the testing process.

- Built-in monitors:

  The component can maintain state, performance load, capacity, security, or other information accessible through an interface. This interface can be a permanent interface of the component or it can be introduced temporarily via an instrumentation technique such as aspect-oriented programming or pre-processor macros. A common technique is to record events when monitoring states have been activated. Monitoring states can actually increase the testing effort since tests may have to be repeated with the monitoring turned off.