Guru Nanak Institute Technical Campus

School of Engineering & Technology

DEPARTMENT OF INFORMATION TECHNOLOGY

CASE TOOLS AND SOFTWARE TESTING Lab Manual

[Subject Code: 57617]

For the Academic Year 2012-2013

IV B Tech Semester -I

Guru Nanak Institute Technical Campus

School of Engineering & Technology

Ibrahimpatnam, R R District -501 506(A.P).

In-charge **HOD** **Principal**

Document No:	Date of Issue:	Compiled by:	Authorized by
GNEC/IT/Lab Manual/CASE TOOLS AND SOFTWARE TESTING [57617]	28-06-2012	Prof. N Prasanna Balaji Mr.P.Hari Shankar Mr.Ch.Ravindra	HOD-IT Prof. N Prasanna

Version 1.3	**Date of Revision** **28-06-2012**	**Verified by:**	Balaji

Guru Nanak Institute Technical Campus

School of Engineering & Technology
Guru Nanak Institute Technical Campus

Ibrahimpatnam, R R District – 501 506 (A. P.)

Department of Information Technology

Lab Manual for the Academic Year 2012-13

(In accordance with JNTU syllabus)

SUBJECT : CASE TOOLS AND SOFTWARE TESTING

SUBJECT CODE : 57617

SEMESTER : I

STREAM : Information Technology

INSTRUCTOR : 1).Mr. P. Hari Shankar

2).Mr. Ch.Ravindra

PROGRAMMER : 1).Ms S. Anjali

2). MsB.Sujani

LAB ASSISTANT : S.Chamkaur Singh

Head IT

INTRODUCTION:

ABOUT LAB:

There are 66 systems (Compaq Presario) installed in this Lab. Their configurations are as follows:

Processor	:	AMD Athelon ™ 1.67 GH_z
RAM	:	256 MB
Hard Disk	:	40 GB
Optical Drive	:	52X CD-ROM
Monitor	:	15" CRT Color Monitor
Mouse	:	Optical Mouse
Key Board	:	105 MMX Key Board
Network Topology	:	Star Topology
Network Interface card	:	Present

Software

1 All systems are configured in **DUAL BOOT** mode i.e, Students can boot from Windows XP or Linux as per their lab requirement.

This is very useful for students because they are familiar with different Operating Systems so that they can execute their programs in different programming environments.

2 Each student has a separate login for database access

Oracle 9i client version is installed in all systems. On the server, account for each student has been created.

This is very useful because students can save their work (scenarios', pl/sql programs, data related projects ,etc) in their own accounts. Each student work is safe and secure from other students.

3 Latest Technologies like **DOT NET** and **J2EE** are installed in some systems. Before submitting their final project, they can start doing mini project from 2nd year onwards.

4 **MASM (Macro Assembler)** is installed in all the systems

Students can execute their assembly language programs using MASM. MASM is very useful students because when they execute their programs

They can see contents of Processor **Registers** and how **each instruction** is being executed in the **CPU**.

1. Rational Rose Software is installed in some systems

   Using this software, students can depict UML diagrams of their projects.

2 Softwares installed: C, C++, JDK1.5, MASM, OFFICE-XP, J2EE and DOT NET, Rational Rose.

3 Systems are provided for students in the 1:1 ratio.

4 Systems are assigned numbers and same system is allotted for students when they do the lab.

GUIDELINES TO STUDENTS:

1 Equipment in the lab for the use of student community. Students need to maintain a proper decorum in the computer lab. Students must use the equipment with care. Any damage is caused is punishable.

2 Students are required to carry their observation / programs book with completed exercises while entering the lab.

3 Students are supposed to occupy the machines allotted to them and are not supposed to talk or make noise in the lab. The allocation is put up on the lab notice board.

4 Lab can be used in free time / lunch hours by the students who need to use the systems should take prior permission from the lab in-charge.

5 Lab records need to be submitted on or before date of submission.

6 Students are not supposed to use floppy disks

LAB CODE

1. Students should come to the lab on time.
2. Students should wear their dress code while coming to the lab
3. Students should bring their records and observation books to the lab compulsorily
4. Students should get their observation book corrected before leaving the lab.
5. Students will be given lab manual which they should return before leaving.
6. Students should do their assigned work in time and submit the work to the lab faculty
7. Students should not bring their cell phones or bags into the lab
8. Equipment in the lab for the use of student community. Students need to maintain a proper decorum in the computer lab. Students must use the equipment with care. Any damage is caused is punishable.
9. Students are supposed to occupy the machines allotted to them and are not supposed to talk or make noise in the lab. The allocation is put up on the lab notice board.

INDEX

1. CASE Tools Introduction:

CASE tools known as Computer-aided software engineering tools is a kind of component-based development which allows its users to rapidly develop information systems. The main goal of case technology is the automation of the entire information systems development life cycle process using a set of integrated software tools, such as modeling, methodology and automatic code generation. Component based manufacturing has several advantages over custom development. The main advantages are the availability of high quality, defect free products at low cost and at a faster time. The prefabricated components are customized as per the requirements of the customers. The components used are pre-built, ready-tested and add value and differentiation by rapid customization to the targeted customers. However the products we get from case tools are only a skeleton of the final product required and a lot of programming must be done by hand to get a fully finished, good product.

Characteristics of CASE:

Some of the characteristics of case tools that make it better than customized development are;

- It is a graphic oriented tool.
- It supports decomposition of process.

Some typical CASE tools are:

- Unified Modeling Language
- Data modeling tools, and
- Source code generation tools

Introduction to UML (Unified Modeling Language):

The UML is a language for specifying, constructing, visualizing, and documenting the software system and its components. The UML is a graphical language with sets of rules and semantics. The rules and semantics of a model are expressed in English in a form known as OCL (Object Constraint Language). OCL uses simple logic for specifying the properties of a system. The UML is not intended to be a visual programming language. However it has a much closer mapping to object-oriented programming languages, so that the best of both can be obtained. The UML is much simpler than other methods preceding it. UML is appropriate for modeling systems, ranging from enterprise information system to distributed web based application and even to real time embedded system. It is a very expensive language addressing all views needed to develop and then to display system even though understand to use. Learning to apply UML effectively starts forming a conceptual mode of languages which requires learning.

Three major language elements:

- UML basic building blocks
- Rules that dictate how this building blocks put together
- Some common mechanism that apply throughout the language

The primary goals in the design of UML are:

1. Provides users ready to use, expressive visual modeling language as well so they can develop and exchange meaningful models.
2. Provide extensibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development processes.
4. Provide formal basis for understanding the modeling language.
5. Encourage the growth of the OO tools market.
6. Support higher-level development concepts.
7. Integrate best practices and methodologies.

Every complex system is best approached through a small set of nearly independent views of a model. Every model can be expressed at different levels of fidelity. The best models are connected to reality. The UML defines nine graphical diagrams:

1. Class diagram

2. Use-case diagram
3. Behavior diagram
    - 3.1. Interaction diagram
        - 3.1.1. sequence diagram
        - 3.1.2. collaboration diagram
    - 3.2. state chart diagram
    - 3.3. activity diagram

4. Implementation diagram
    - 4.1 component diagram
    - 4.2 deployment diagram

1. UML class diagram:

The UML class diagram is also known as object modeling. It is a static analysis diagram. These diagrams show the static structure of the model. A class diagram is a connection of static model elements, such as classes and their relationships, connected as a graph to each other and to their contents.

2. Use-case diagram:

The functionality of a system can be described in a number of different use-cases, each of which represents a specific flow of events in a system. It is a graph of actors, a set of use-cases enclosed in a boundary, communication, associations between the actors and the use-cases, and generalization among the use-cases.

3. Behavior diagram:

It is a dynamic model unlike all the others mentioned before. The objects of an object oriented system are not static and are not easily understood by static diagrams. The behavior of the class's instance (an object) is represented in this diagram. Every use-case of the system has an associated behavior diagram that indicates the behavior of the object. In conjunction with the use-case diagram we may provide a script or interaction diagram to show a time line of events. It consists of sequence and collaboration diagrams.

4. Interaction diagram

It is the combination of sequence and collaboration diagram. It is used to depict the flow of events in the system over a timeline. The interaction diagram is a dynamic model which shows how the system behaves during dynamic execution.

5. State chart diagram:

It consists of state, events and activities. State diagrams are a familiar technique to describe the behavior of a system. They describe all of the possible states that a particular object can get into and how the object's state changes as a result of events that reach the object. In most OO techniques, state diagrams are drawn for a single class to show the lifetime behavior of a single object.

6. Activity diagram:

It shows organization and their dependence among the set of components. These diagrams are particularly useful in connection with workflow and in describing behavior that has a lot of parallel processing. An activity is a state of doing something: either a real-world process, or the execution of a software routine.

7. Implementation diagram:

It shows the implementation phase of the systems development, such as the source code structure and the run-time implementation structure. These are relatively simple high level diagrams compared to the others seen so far. They are of two sub-diagrams, the component diagram and the deployment diagram.

8. Component diagram:

These are organizational parts of a UML model. These are boxes to which a model can be decomposed. They show the structure of the code itself. They model the physical components such as source code, user interface in a design. It is similar to the concept of packages.

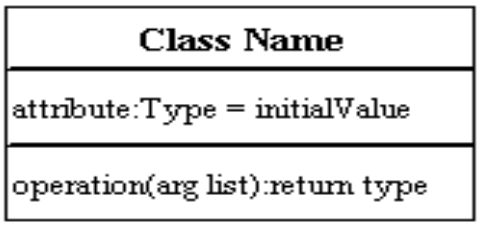
9. Deployment diagram:

The deployment diagram shows the structure of the runtime system. It shows the configuration of runtime processing elements and the software components that live in them. They are usually used in

conjunction with deployment diagrams to show how physical modules of code are distributed on the system.

2. UML NOTATION

Classes

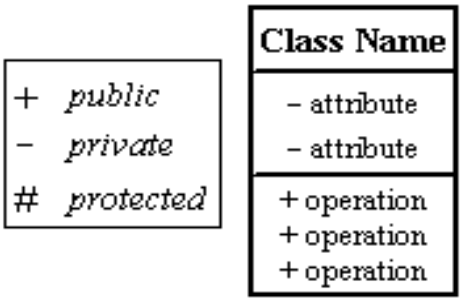


Illustrate classes with rectangles divided into compartments. Place the name of the class in the first partition (centered, bolded, and capitalized), list the attributes in the second partition, and write operations into the third.

Active Class

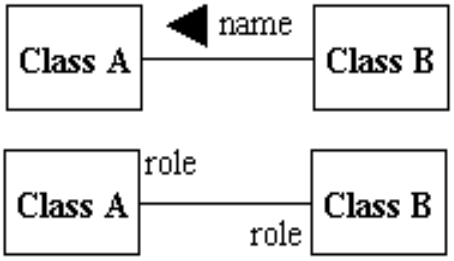
Active classes initiate and control the flow of activity, while passive classes store data and serve other classes. Illustrate active classes with a thicker border.

Visibility



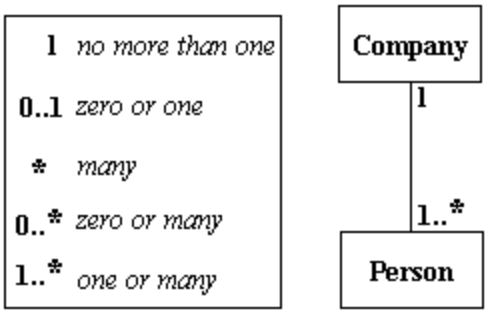
Use visibility markers to signify who can access the information contained within a class. Private visibility hides information from anything outside the class partition. Public visibility allows all other classes to view the marked information. Protected visibility allows child classes to access information they inherited from a parent class.

Associations



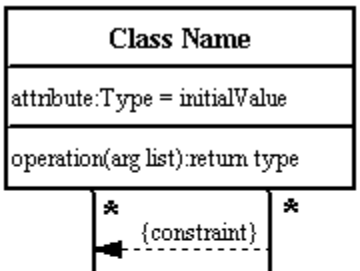
Associations represent static relationships between classes. Place association names above, on, or below the association line. Use a filled arrow to indicate the direction of the relationship. Place roles near the end of an association. Roles represent the way the two classes see each other. ***Note:***It's uncommon to name both the association and the class roles.

Multiplicity (Cardinality)



Place multiplicity notations near the ends of an association. These symbols indicate the number of instances of one class linked to *one* instance of the other class. For example, one company will have one or more employees, but each employee works for one company only.

Constraint



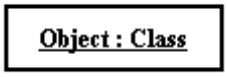
Place constraints inside curly braces {}.

Associations in which an object is part of a whole are aggregations. **Composition** is a strong association in which the part can belong to only one whole -- the part cannot exist without the whole. Composition is denoted by a filled diamond at the whole end.

Symbol	Access
+	public
-	private
#	protected

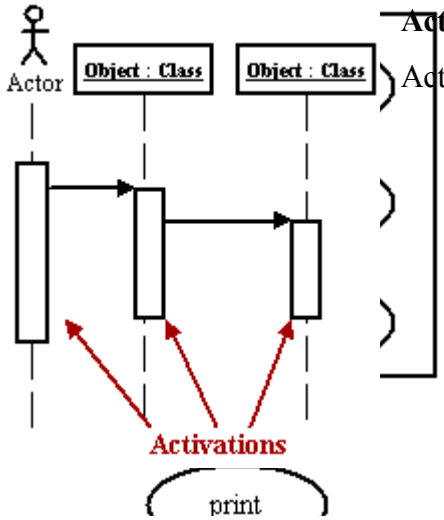
BASIC USE CASE DIAGRAM SYMBOLS AND NOTATIONS

Class roles



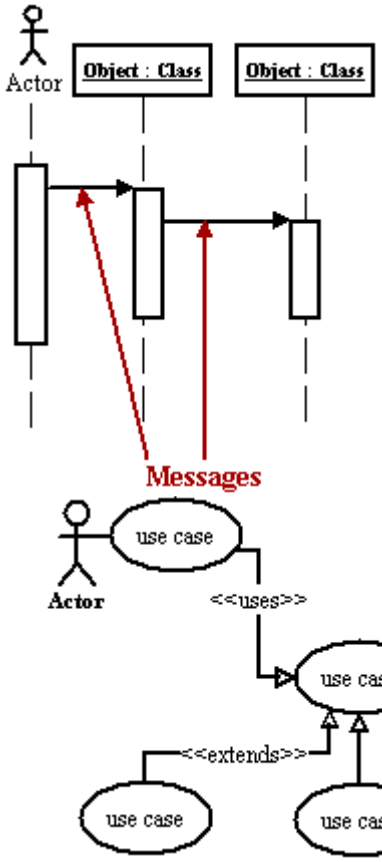
Class roles describe the way an object will behave in context. Use the UML object symbol to illustrate class roles, but don't list object attributes.

Activations



Activation boxes represent the time an object needs to complete a task.

Messages



Messages are arrows that represent communication between objects. Use half-arrowed lines to represent asynchronous messages. Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its tasks.

Actors

Actors are the users of a system. When one system is the actor of another system, label the actor system with the actor stereotype.

Relationships

Illustrate relationships between an actor and a use case with a simple line. For relationships among use cases, use s" or "extends." A "uses" use case is needed by another . An "extends" relationship under a certain use case.

Arrow	Message type
→	Simple
→	Synchronous
→	Asynchronous
↩	Balking
→	Time out

System

Draw your system's boundaries using a rectangle that contains use cases. Place actors outside the system's boundaries.

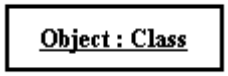
Use Case

Draw use cases using ovals. Label with ovals with verbs that represent the system's functions.

Various message types for Sequence and Collaboration diagrams

BASIC COLLABORATION DIAGRAM SYMBOLS AND NOTATIONS

Class roles

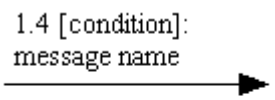


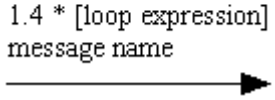
Class roles describe how objects behave. Use the UML object symbol to illustrate class roles, but don't list object attributes.

Association roles

Association roles describe how an association will behave given a particular situation. You can draw association roles using simple lines labeled with stereotypes.

Messages



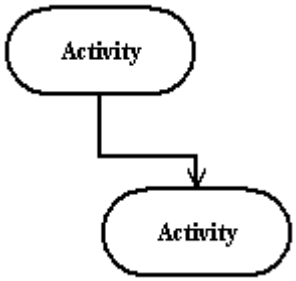


Unlike sequence diagrams, collaboration diagrams do not have an explicit way to denote time and instead number messages in order of execution. Sequence numbering can become nested using the Dewey decimal system. For example, nested messages under the first message are labeled 1.1, 1.2, 1.3, and so on. The condition for a message is usually placed in square brackets immediately following the sequence number. Use a * after the sequence number to indicate a loop.

BASIC ACTIVITY DIAGRAM SYMBOLS AND NOTATIONS

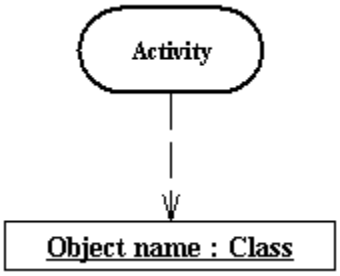
Action states

Action states represent the no interruptible actions of objects. You can draw an action state in Smart Draw using a rectangle with rounded corners.



Action Flow

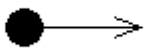
Action flow arrows illustrate the relationships among action states.



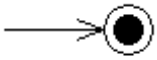
Object Flow

Object flow refers to the creation and modification of objects by activities. An object flow arrow from an action to an object means that the action creates or influences the object. An object flow arrow from an object to an action indicates that the action state uses the object.

Initial State

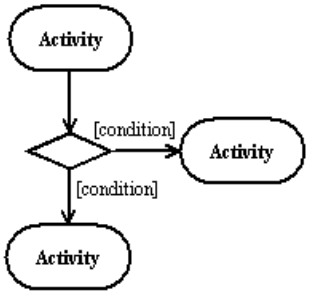


A filled circle followed by an arrow represents the initial action state.



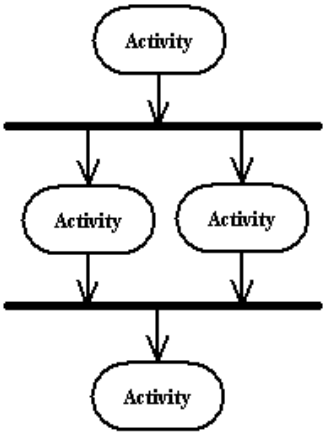
Final State

An arrow pointing to a filled circle nested inside another circle represents the final action state.



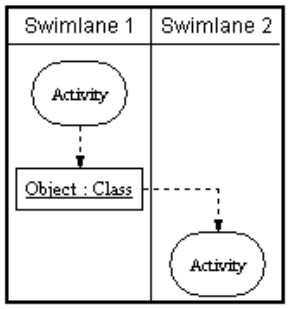
Branching

A diamond represents a decision with alternate paths. The outgoing alternates should be labeled with a condition or guard expression. You can also label one of the paths "else."



Synchronization

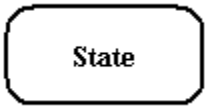
A synchronization bar helps illustrates parallel transitions. Synchronization is also called forking and joining.



Swim lanes

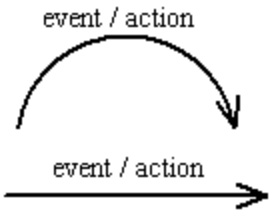
Swim lanes group related activities into one column.

States



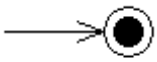
States represent situations during the life of an object. You can easily illustrate a state in Smart Draw by using a rectangle with rounded corners.

Transition



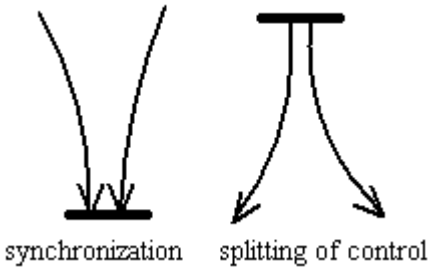
A solid arrow represents the path between different states of an object. Label the transition with the event that triggered it and the action that results from it.

Final State



An arrow pointing to a filled circle nested inside another circle represents the object's final state.

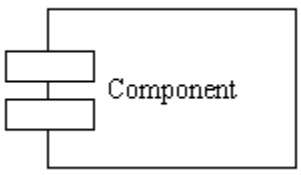
Synchronization and Splitting of Control



A short heavy bar with two transitions entering it represents a synchronization of control. A short heavy bar with two transitions leaving it represents a splitting of control that creates multiple states.

BASIC COMPONENT DIAGRAM SYMBOLS AND NOTATIONS

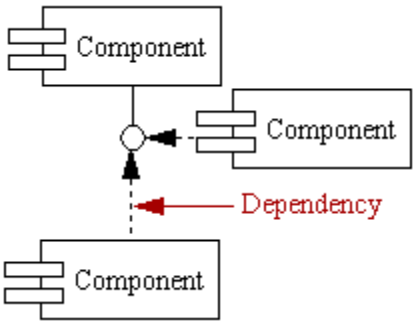
Component



A component is a physical building block of the system. It is represented as a rectangle with tabs.

Interface

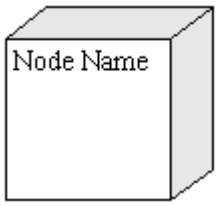
An interface describes a group of operations used or created by components.



Dependencies

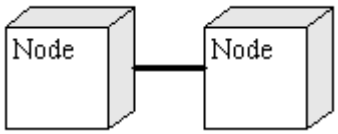
Draw dependencies among components using dashed arrows.

BASIC DEPLOYMENT DIAGRAM SYMBOLS AND NOTATIONS



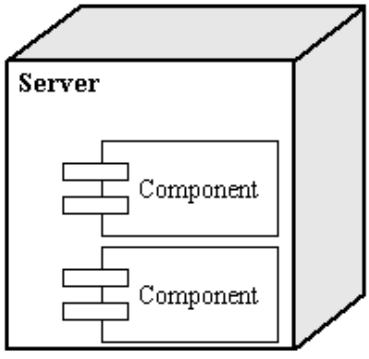
Node

A node is a physical resource that executes code components.



Association

Association refers to a physical connection between nodes, such as Ethernet.



Components and Nodes

Place components inside the node that deploys them.

ATM SYSTEM

Aim:

To create a system to perform Bank ATM transaction

Theory

Problem analysis and project planning

Introduction

Banking is one of the common and day to day attribute of life. Nowadays it is totally different from that existed a few years ago banking has become completely computerized new facilities such as credit cards, debit cards & ATM has been introduced. ATM is automatic teller machine which is basically used to withdraw money from an account.

Objectives

The objective of this software is similar to ATM software installed in ATM center. It should first validate the pin in the ATM card. Then the type of transaction is enquired and the information from the customer is validated. If it is a withdrawal the amount is asked. After the money is delivered the transaction just made is updated in the database where the customer's information is stored.

Scope

The scope of the project is to design an ATM system that will help in completely automatic banking this software is going to be designed for withdrawal and deposit of money and register the transaction in the database where the customer's information is stored.

Problem Statement

ATM is another type of banking where the most frequently type of transaction made is withdrawal. A user may withdraw as much as many amount as he wants until his account holds a sum

greater than his withdrawal amount. ATM is completely automated and there is no necessity of the ATM center being placed at the bank itself. It can be placed in the shopping malls, airports, railway stations etc.

This ATM system can use any kind of interface. But it should be user friendly and not confusing. Help manuals should be provided in case any customer has problem working with the software.

The system will retain information on the entire customer who has necessity rights to access the service. It will contain the balance amount in the account, rate of interest, any special allowance for that customer and most of all pin number of the customer. The ATM system should be compatible with any kind of database such as MS-ACCESS, DB2, ORACLE, SQL, SERVER etc. the emphasis here is on consistency.

Some customer could have availed some special offers on his ATM cards. So this must be taken care of and the appropriate data should be dealt with.

The ATM should provide easy access to the data for the customer. It should also have a highly secure interface so that one can take money one behalf of others. So the security is one of the main aspects in ATM.

USE-CASE diagram:

The ATM transaction use cases in our system are:

1. Login
2. Withdraw
3. Mini statement
4. ATM machine status
5. Deposit

Actors involved:

1. User
2. Bank manager

USE-CASE name: Login

The user enters a user name and password. If it is valid, the user's account becomes available. If it is invalid, an appropriate message is displayed to the user.

USE-CASE name: Withdraw

The user tries to withdraw an amount from his or her checking account. The amount is less than or equal to the checking account's balance, the transaction is performed and the available information is displayed. The system creates a record of the transaction and the display confirmation message is displayed to the client.

USE-CASE name: Mini statement

The bank user requests a history of transactions for a checking account. The system displays the transaction history for the checking account. The transaction history consists of amount, date, transaction type and balance of the particular account.

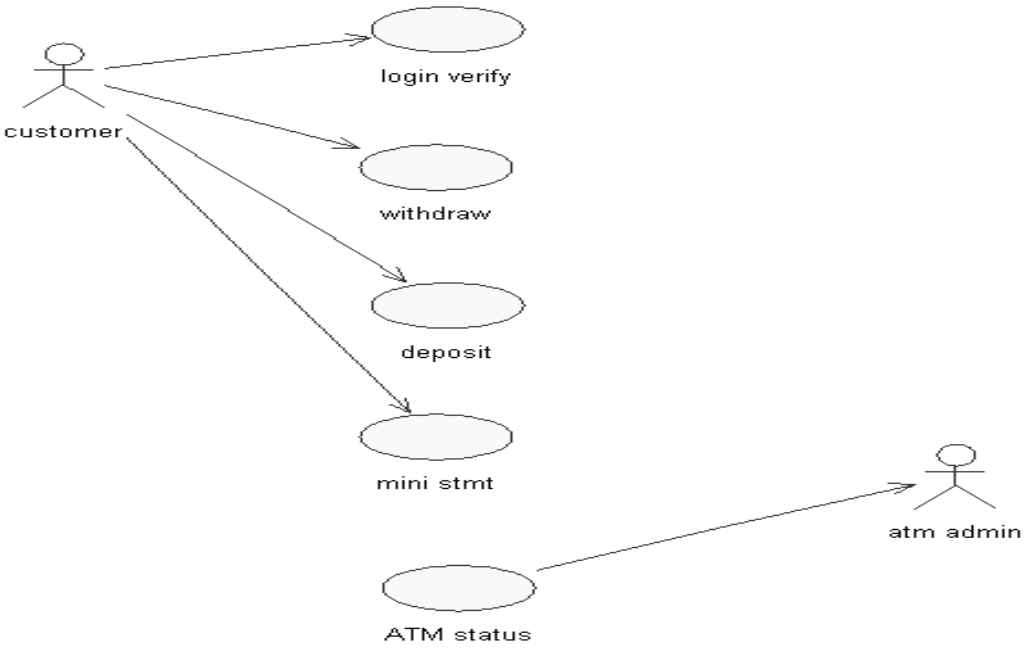
USE-CASE name: ATM machine status

The bank manager enters a username and password. If it is valid, the bank manager accesses the machine status. If it is invalid, an appropriate message is displayed to the user.

USE-CASE name: Deposit

The bank user requests the system to deposit money to an account. The user accesses the account for which a deposit is going to be made and enters the amount. The system creates a record of the transaction and an appropriate confirmation message (display confirmation) is displayed to the client. The transaction must include the date, type, amount and account balance after the transaction.

Use-case diagram for ATM system



Class diagram

The class diagram, also referred to as object modeling is the main static analysis diagram. The main task of object modeling is to graphically show what each object will do in the problem domain. The problem domain describes the structure and the relationships among objects.

The ATM system class diagram consists of four classes:

1. User class
2. ATM machine status
3. Account

4. Transaction

1) User class:

It consists of four attributes and two operations. The attributes are user name, password, address and DOB. The operations of this class are read (), display () and write ().

2) ATM machine status:

The attributes of this class are ATM balance, today's withdrawal, today's balance, and limitations. The operations are login verification (), ATM status () and display confirmation ().

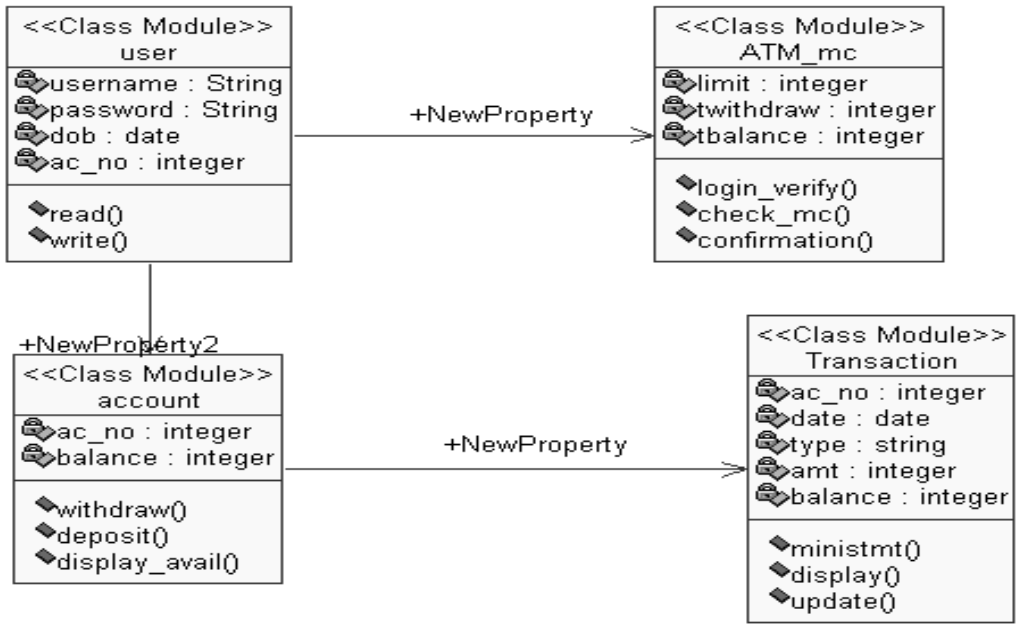
3) Account:

The attributes are account no. and balance and the operations are withdraw (), deposit () and display availability ().

4) Transaction:

The attributes of this class are account no, transaction type, data, amount, balance and the operations are mini statement () and create transaction ().

Class diagram for ATM system



Sequence diagram:

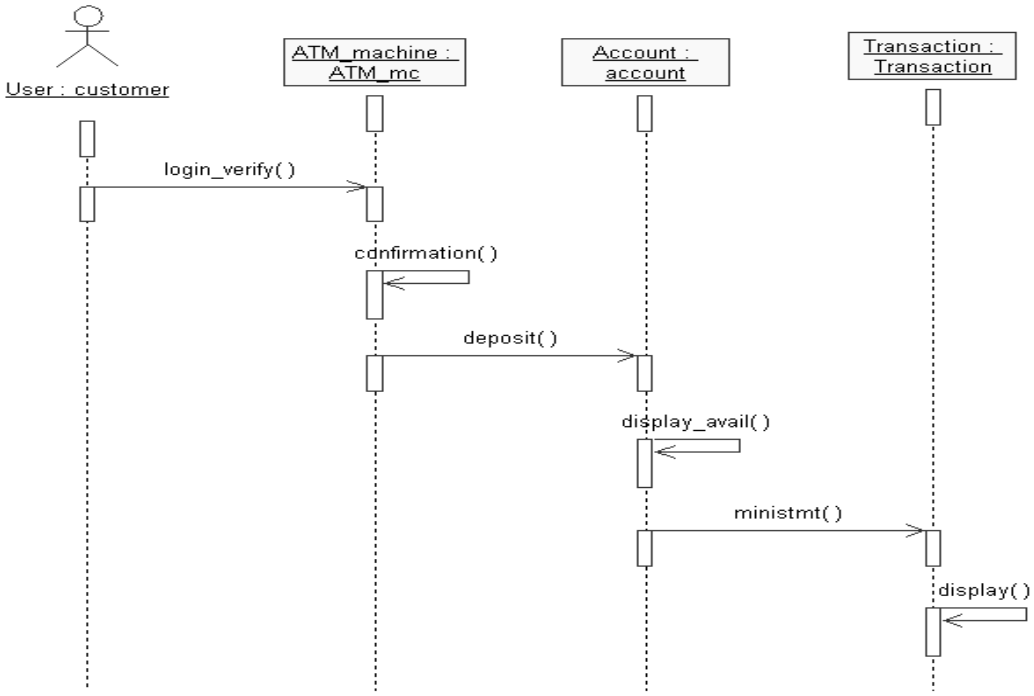
A sequence diagram represents the sequence and interactions of a given USE-CASE or scenario. Sequence diagrams can capture most of the information about the system. Most object to object interactions and operations are considered events and events include signals, inputs, decisions, interrupts, transitions and actions to or from users or external devices.

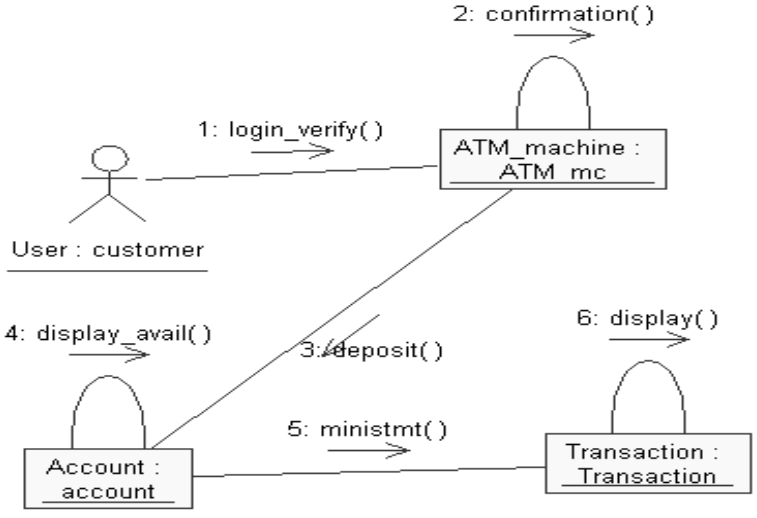
An event also is considered to be any action by an object that sends information. The event line represents a message sent from one object to another, in which the "form" object is requesting an operation be performed by the "to" object. The "to" object performs the operation using a method that the class contains.

It is also represented by the order in which things occur and how the objects in the system send message to one another.

The sequence diagram for each USE-CASE that exists when a user withdraws, deposits, needs information about ATM machine status and account are drowned.

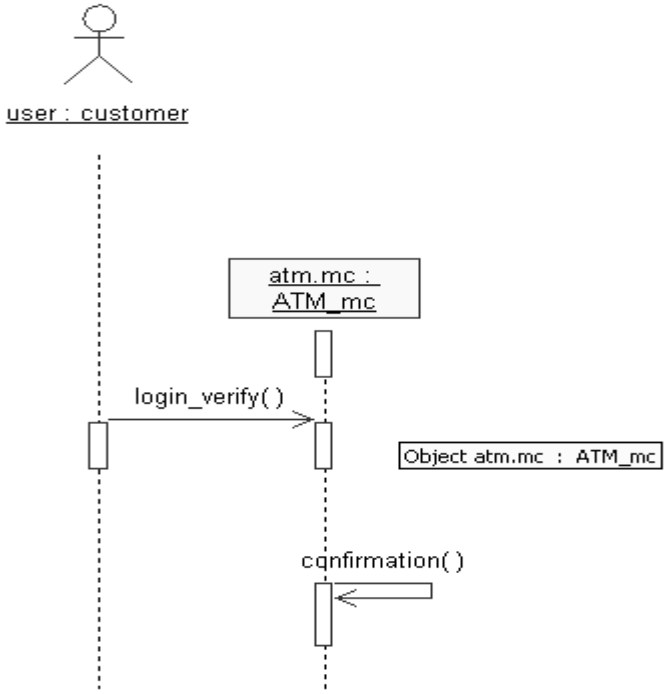
Sequence and collaboration diagram for deposit process

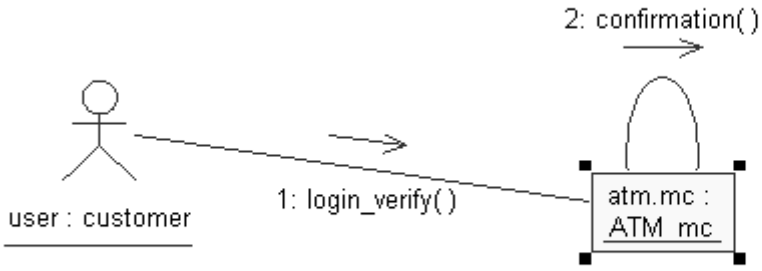




The diagrams show the entire deposit process in an ATM system. The user has to login to the ATM machine and deposit the amount of money as required by the user. The user may wish to get a mini-statement and a screen about the details of the transaction.

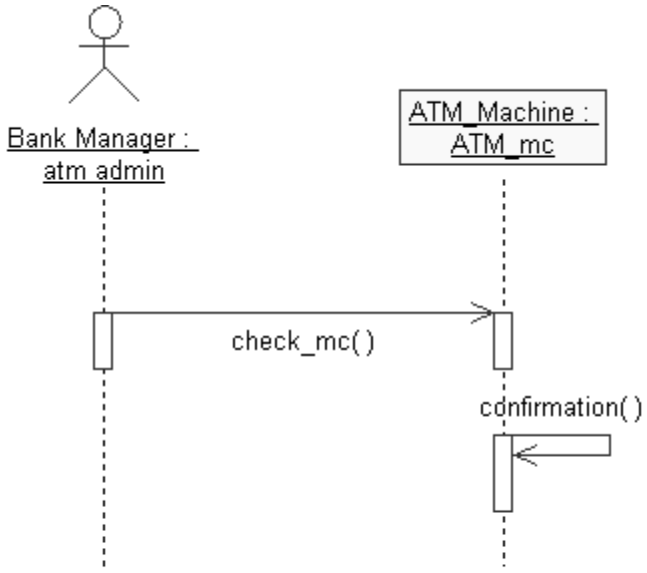
Sequence and collaboration diagram for login

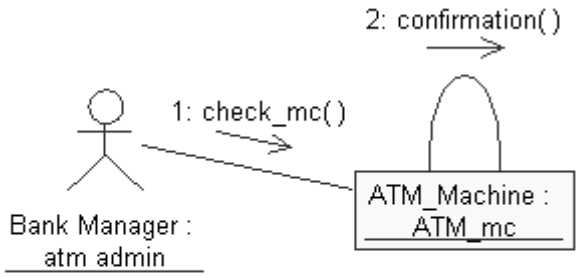




The diagrams show the process of login by the user to the ATM system. The user has to enter his details. The details entered are verified by the system and the user is approved if the details match, otherwise an appropriate error message is displayed.

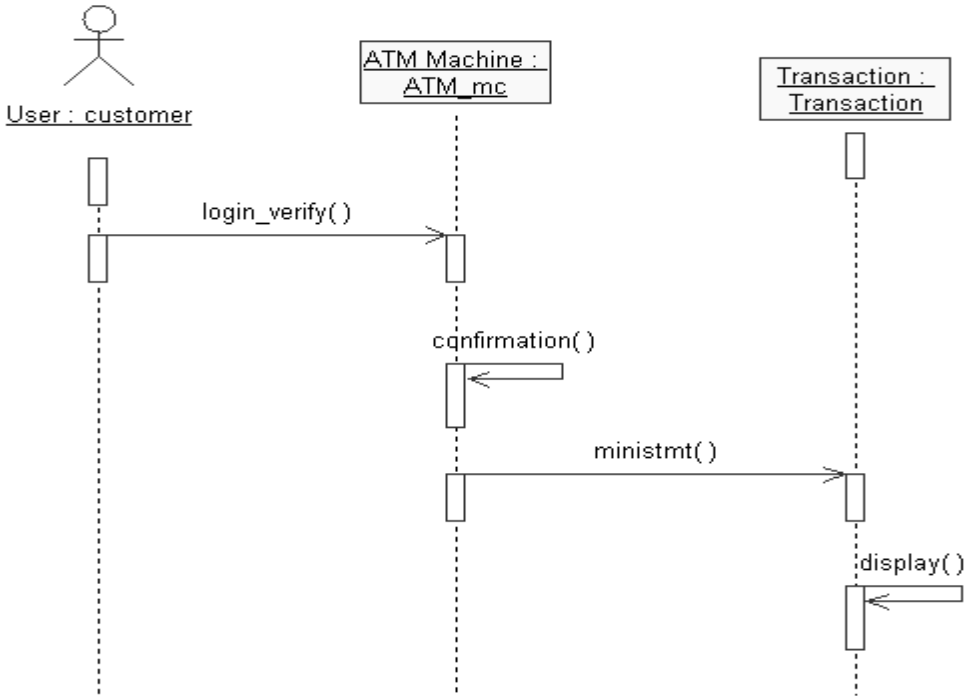
Sequence and collaboration diagram for checking machine status

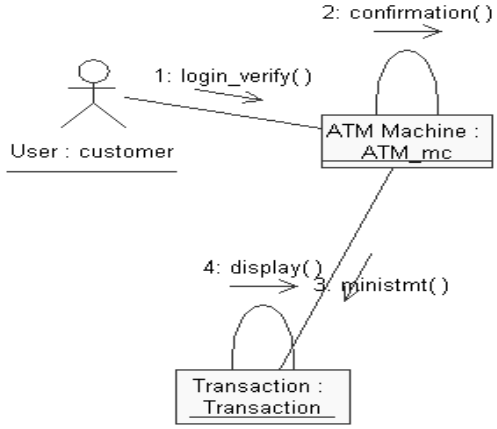




The Administrator of the ATM system has to maintain the details about the ATM, He has to check if there is enough money in the ATM and if the ATM is functional without major errors. For this, he may check the ATM machine status occasionally. The process is shown in the above diagrams.

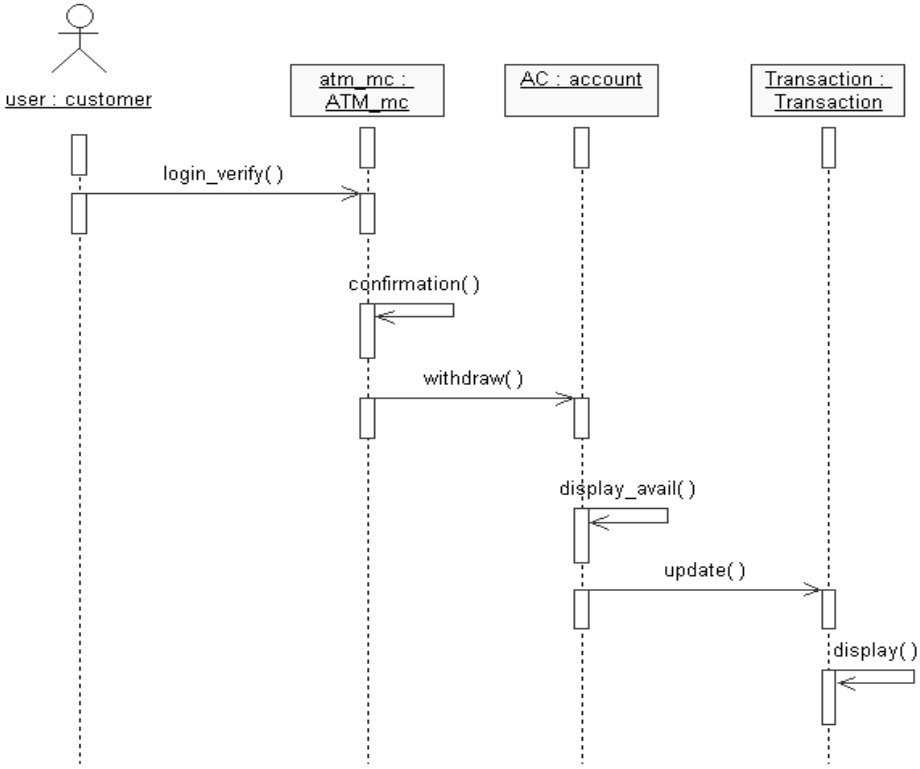
Sequence and collaboration diagram for printing mini statement

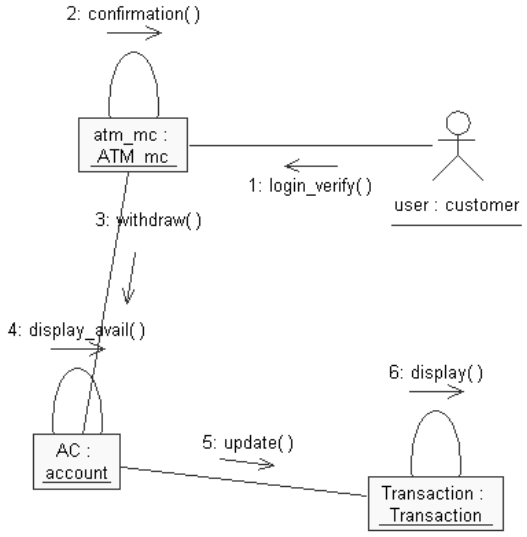




After a transaction is carried out successfully, the user must get a mini-statement to tell him his account's details such as balance and transaction number. This process is depicted in the above diagrams.

Sequence and collaboration diagram for withdraw process

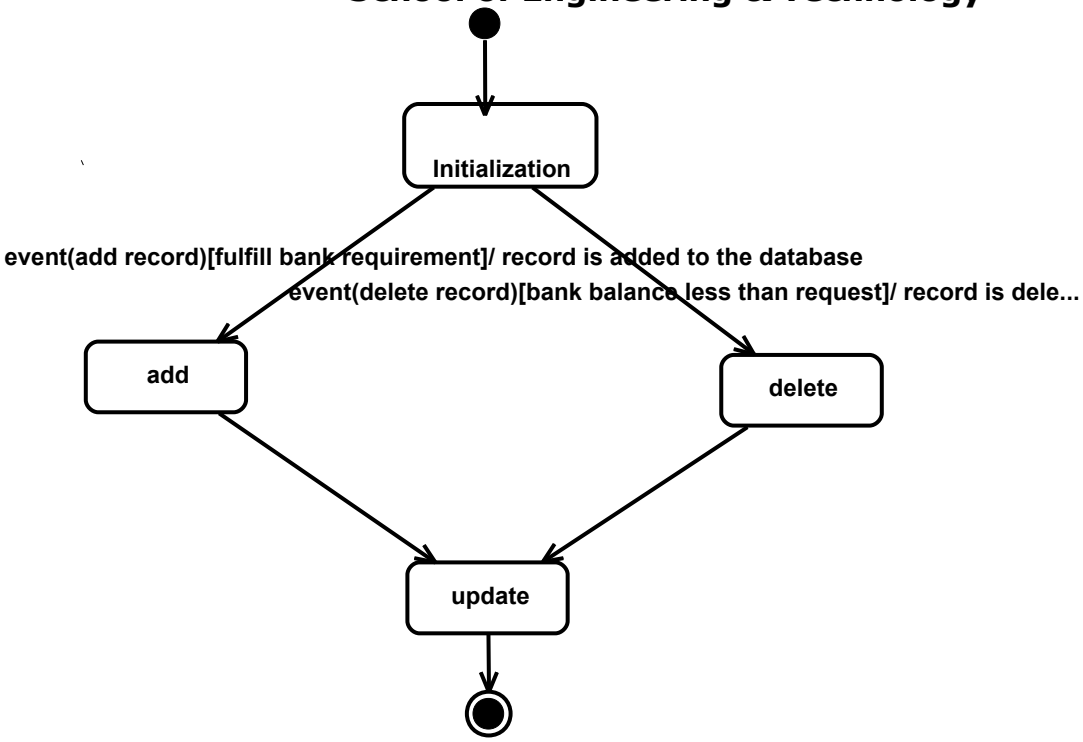




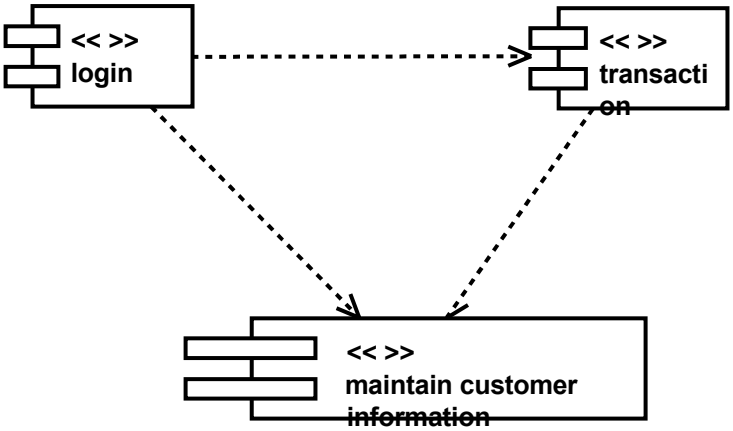
The user can make withdraw money from his account. The process is depicted in the diagrams above. The user has to login to the system using his username and password, which are verified by the

system. After successful verification, the user can choose the amount of money he wants to withdraw from his account. The amount specified by the user is checked by the system to make sure there is enough balance in his account to carry out the transaction. After the transaction is carried out the resulting amount is displayed and the details are updated to the database.

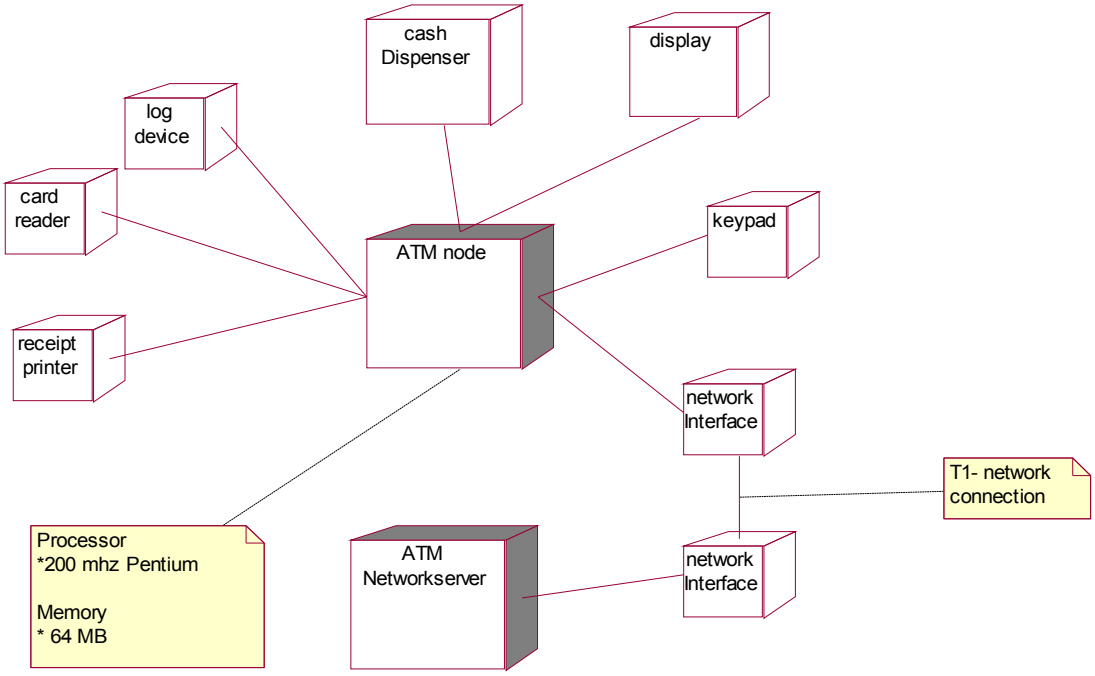
State Chart diagram for ATM system



Component diagram for ATM system



Deployment Diagram for ATM system



Experiment 1: Study of Any Testing Tool (Win Runner)

Win Runner is a program that is responsible for the automated testing of software. Win Runner is a Mercury Imperative's enterprise functional testing tool for Microsoft windows applications.

Importance of Automated Testing:

1. Reduced testing time
2. Consistent test procedures – ensure process repeatability and resource independence. Eliminates errors of manual testing
3. Reduces QA cost – Upfront cost of automated testing is easily recovered over the lifetime of the product
4. Improved testing productivity – test suites can be run earlier and more often
5. Proof of adequate testing
6. For doing tedious work – test team members can focus on quality areas.

Win Runner Uses:

1. With Win Runner sophisticated automated tests can be created and run on an application.
2. A series of wizards will be provided to the user, and these wizards can create tests in an automated manner.
3. Another impressive aspect of Win Runner is the ability to record various interactions, and transform them into scripts. Win Runner is designed for testing graphical user interfaces.
4. When the user makes an interaction with the GUI, this interaction can be recorded. Recording the interactions allows determining various bugs that need to be fixed.
5. When the test is completed, Win Runner will provide with detailed information regarding the results. It will show the errors that were found, and it will also give important information about them. The good news about these tests is that they can be reused many times.
6. Win Runner will test the computer program in a way that is very similar to normal user interactions. This is important, because it ensures a high level of accuracy and realism. Even if an

engineer is not physically present, the Recover manager will troubleshoot any problems that may occur, and this will allow the tests to be completed without errors.

7. The Recover Manager is a powerful tool that can assist users with various scenarios. This is important, especially when important data needs to be recovered.

The goal of Win Runner is to make sure business processes are properly carried out. Win Runner uses TSL, or Test Script Language.

Win Runner Testing Modes

Context Sensitive

Context Sensitive mode records your actions on the application being tested in terms of the GUI objects you select (such as windows, lists, and buttons), while ignoring the physical location of the object on the screen. Every time you perform an operation on the application being tested, a TSL statement describing the object selected and the action performed is generated in the test script. As you record, Win Runner writes a unique description of each selected object to a GUI map.

The GUI map consists of files maintained separately from your test scripts. If the user interfaces of your application changes, you have to update only the GUI map, instead of hundreds of tests. This allows you to easily reuse your Context Sensitive test scripts on future versions of your application.

To run a test, you simply play back the test script. Win Runner emulates a user by moving the mouse pointer over your application, selecting objects, and entering keyboard input. Win Runner reads the object descriptions in the GUI map and then searches in the application being tested for objects matching these descriptions. It can locate objects in a window even if their placement has changed.

Analog

Analog mode records mouse clicks, keyboard input, and the exact x- and y-coordinates traveled by the mouse. When the test is run, Win Runner retraces the mouse tracks. Use Analog mode when exact mouse coordinates are important to your test, such as when testing a drawing application.

The Win Runner Testing Process

Testing with *Win Runner* involves six main stages:

1. Create the GUI Map

The first stage is to create the GUI map so Win Runner can recognize the GUI objects in the application being tested. Use the Rapid Test Script wizard to review the user interface of your

application and systematically add descriptions of every GUI object to the GUI map. Alternatively, you can add descriptions of individual objects to the GUI map by clicking objects while recording a test.

2. Create Tests

Next is creation of test scripts by recording, programming, or a combination

of both. While recording tests, insert checkpoints where we want to check the response of the application being tested. We can insert checkpoints that check GUI objects, bitmaps, and databases. During this process, Win Runner captures data and saves it as *expected results*—the expected response of the application being tested.

3. Debug Tests

Run tests in Debug mode to make sure they run smoothly. One can set breakpoints, monitor variables, and control how tests are run to identify and isolate defects. Test results are saved in the debug folder, which can be discarded once debugging is finished.

When Win Runner runs a test, it checks each script line for basic syntax errors, like incorrect syntax or missing elements in **If**, **While**, **Switch**, and **For** statements. We can use the **Syntax Check** options (**Tools >Syntax Check**) to check for these types of syntax errors before running your test.

4. Run Tests

Tests can be run in Verify mode to test the application. Each time Win Runner encounters a checkpoint in the test script, it compares the current data of the application being tested to the expected data captured earlier. If any mismatches are found, Win Runner captures them as *actual results*.

5. View Results

Following each test run, Win Runner displays the results in a report. The report details all the major events that occurred during the run, such as checkpoints, error messages, system messages, or user messages.

If mismatches are detected at checkpoints during the test run, we can view the expected results and the actual results from the Test Results window. In cases of bitmap mismatches, one can also view a bitmap that displays only the difference between the expected and actual results.

We can view results in the standard Win Runner report view or in the Unified report view. The Win Runner report view displays the test results in a Windows-style viewer. The Unified report view displays the results in an HTML-style viewer (identical to the style used for Quick Test Professional test results).

6. Report Defects

If a test run fails due to a defect in the application being tested, one can report information about the defect directly from the Test Results window.

This information is sent via e-mail to the quality assurance manager, who tracks the defect until it is fixed.

Using Win Runner Window

Before you begin creating tests, you should familiarize yourself with the Win Runner main window.

To start Win Runner:

Choose **Programs** > **Win Runner** > **Win Runner** on the **Start** menu.

The first time you start Win Runner, the Welcome to Win Runner window and the "What's New in Win Runner" help open. From the Welcome window you can create a new test, open an existing test, or view an overview of Win Runner in your default browser.

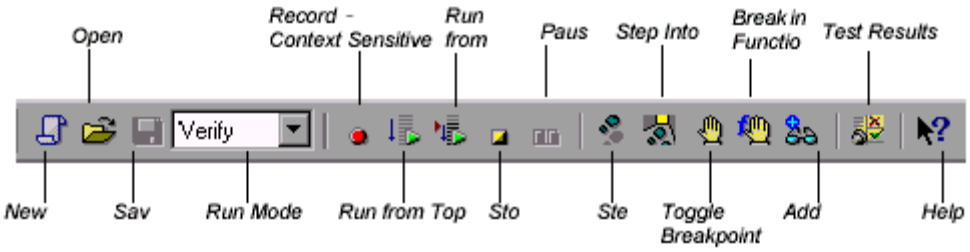
If you do not want this window to appear the next time you start Win Runner, clear the **Show on Startup** check box. To show the **Welcome to Win Runner** window upon startup from within Win Runner, choose **Settings > General Options**, click the **Environment** tab, and select the **Show Welcome screen** check box.

The Main Win Runner Window

The main Win Runner window contains the following key elements:

- *Win Runner title bar*
- *Menu bar,* with drop-down menus of Win Runner commands
- *Standard toolbar,* with buttons of commands commonly used when running a test
- *User toolbar,* with commands commonly used while creating a test
- *Status bar*, with information on the current command, the line number of the insertion point and the name of the current results folder

The *Standard toolbar* provides easy access to frequently performed tasks, such as opening, executing, and saving tests, and viewing test results.



Standard Toolbar

The *User toolbar* displays the tools you frequently use to create test scripts. By default, the User toolbar is hidden. To display the User toolbar, choose **Window** > **User Toolbar**. When you create tests, you can minimize the Win Runner window and work exclusively from the toolbar.

The User toolbar is customizable. You choose to add or remove buttons using the **Settings > Customize User Toolbar** menu option. When you re-open Win Runner, the User toolbar appears as it was when you last closed it.

The commands on the Standard toolbar and the User toolbar are described in detail in subsequent lessons.

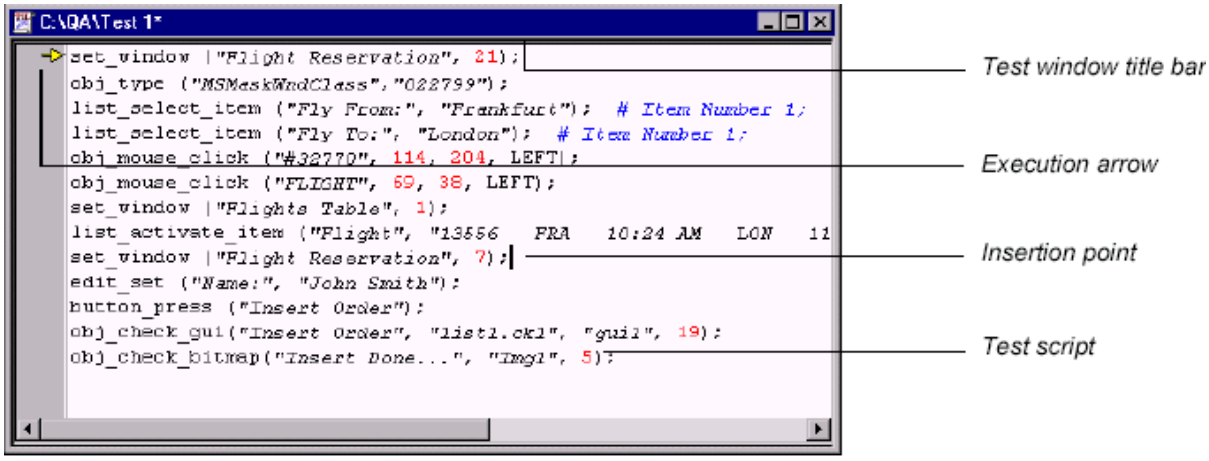
Note that you can also execute many commands using *soft keys*. Soft keys are keyboard shortcuts for carrying out menu commands. You can configure the soft key combinations for your keyboard using the Soft key Configuration utility in your Win Runner program group. For more information, see the "Win Runner at a Glance" chapter in your *Win Runner User's Guide*.

Now that you are familiar with the main Win Runner window, take a few minutes to explore these window components before proceeding to the next lesson.

The Test Window

You create and run Win Runner tests in the test window. It contains the following key elements:

- *Test window title bar*, with the name of the open test
- *Test script*, with statements generated by recording and/or programming in TSL, Mercury Interactive Test Script Language
- *Execution arrow,* which indicates the line of the test script being executed during a test run, or the line that will next run if you select the Run from arrow option
- *Insertion point,* which indicates where you can insert or edit text



Experiment: Study of any web testing tool (e.g. Selenium)

Selenium is a robust set of tools that supports rapid development of test automation for web-based applications. Selenium provides a rich set of testing functions specifically geared to the needs of testing of a web application. These operations are highly flexible, allowing many options for locating UI elements and comparing expected test results against actual application behavior. One of Selenium's key features is the support for executing one's tests on multiple browser platforms.

Selenium Components

Selenium is composed of three major tools. Each one has a specific role in aiding the development of web application test automation.

Selenium-IDE

Selenium-IDE is the Integrated Development Environment for building Selenium test cases. It operates as a Fire fox add-on and provides an easy-to-use interface for developing and running individual test cases or entire test suites. Selenium-IDE has a recording feature, which will keep account of user actions as they are performed and store them as a reusable script to play back. It also has a context menu (right-click) integrated with the Fire fox browser, which allows the user to pick from a list of assertions and verifications for the selected location. Selenium-IDE also offers full editing of test cases for more precision and control.

Although Selenium-IDE is a Fire fox only add-on, tests created in it can also be run against other browsers by using Selenium-RC and specifying the name of the test suite on the command line.

Selenium-RC (Remote Control)

Selenium-RC allows the test automation developer to use a programming language for maximum flexibility and extensibility in developing test logic. For instance, if the application under test returns a result set, and if the automated test program needs to run tests on each element in the result set, the programming language's iteration support can be used to iterate through the result set, calling Selenium commands to run tests on each item.

Selenium-RC provides an API (Application Programming Interface) and library for each of its supported languages: HTML, Java, C#, Perl, PHP, Python, and Ruby. This ability to use Selenium-RC with a high-level programming language to develop test cases also allows the automated testing to be integrated with a project's automated build environment.

Selenium-Grid

Selenium-Grid allows the Selenium-RC solution to scale for large test suites or test suites that must be run in multiple environments. With Selenium-Grid, multiple instances of Selenium-RC are running on various operating system and browser configurations; Each of these when launching register with a hub. When tests are sent to the hub they are then redirected to an available Selenium-RC, which will launch the browser and run the test. This allows for running tests in parallel, with the entire test suite theoretically taking only as long to run as the longest individual test.

* Tests developed on Fire fox via Selenium-IDE can be executed on any other supported browser via a simple Selenium-RC command line.

** Selenium-RC server can start any executable, but depending on browser security settings there may be technical limitations that would limit certain features.

Flexibility and Extensibility

Selenium is highly flexible. There are multiple ways in which one can add functionality to Selenium's framework to customize test automation for one's specific testing needs. This is, perhaps, Selenium's strongest characteristic when compared with proprietary test automation tools and other open source solutions. Selenium-RC support for multiple programming and scripting languages allows the test writer to build any logic they need into their automated testing and to use a preferred programming or scripting language of one's choice.

Selenium-IDE allows for the addition of user-defined "user-extensions" for creating additional commands customized to the user's needs. Also, it is possible to re-configure how the Selenium-IDE generates its Selenium-RC code. This allows users to customize the generated code to fit in with their own test frameworks. Finally, Selenium is an Open Source project where code can be modified and enhancements can be submitted for contribution.

.**Test Suites**

A test suite is a collection of tests. Often one will run all the tests in a test suite as one continuous batch-job.

When using Selenium-IDE, test suites also can be defined using a simple HTML file. The syntax again is simple. An HTML table defines a list of tests where each row defines the file system path to each test. An example tells it all.

```
<html> <head>

<title>Test Suite Function Tests - Priority 1</title> </head>

<body>

<table>

  <tr><td><b>Suite Of Tests</b></td></tr>

  <tr><td><a href="./Login.html">Login</a></td></tr>

  <tr><td><a href="./SearchValues.html">Test Searching for Values</a></td></tr>

  <tr><td><a href="./SaveValues.html">Test Save</a></td></tr>

</table> </body>

</html>
```

A file similar to this would allow running the tests all at once, one after another, from the Selenium-IDE.

Test suites can also be maintained when using Selenium-RC. This is done via programming and can be done a number of ways. Commonly J-unit is used to maintain a test suite if one is using Selenium-RC with Java. Additionally, if C# is the chosen language, N-unit could be employed. If using an interpreted language like Python with Selenium-RC than some simple programming would be involved in setting up a test suite. Since the whole reason for using Sel-RC is to make use of programming logic for your testing this usually isn't a problem.

Few typical Selenium commands:

open - opens a page using a URL.

click/clickAndWait - performs a click operation, and waits for a new page to load.

verifyTitle/assertTitle - verifies an expected page title.

verifyTextPresent - verifies expected text is somewhere on the page.

verifyElementPresent - verifies an expected UI element, is present on the page.

verifyText - verifies expected text and it's corresponding HTML tag are present on the page.

verifyTable - verifies a table's expected contents.

waitForPageToLoad - pauses execution until an expected new page loads.

waitForElementPresent - pauses execution until an expected UI element, is present on the page.

Experiment: Study of Any Bug Tracking Tool (Bugzilla, Bugbit)

Bugzilla is a "Bug Tracking System" that can efficiently keep track of outstanding bugs in a product. Multiple users can access this database and query, add and manage these bugs. Bugzilla essentially comes to the rescue of a group of people working together on a product as it enables them to view current bugs and make contributions to resolve issues.

Its basic repository nature works out better than the mailing list concept and an organized database is always easier to work with.

Advantage of Using Bugzilla:

1. Bugzilla is very adaptable to various situations. Known uses currently include IT support queues, Systems Administration deployment management, chip design and development problem tracking (both pre-and-post fabrication), and software and hardware bug tracking for luminaries such as Red hat, NASA, Linux-Mandrake, and VA Systems. Combined with systems such as CVS, Bugzilla provides a powerful, easy-to-use solution to configuration management and replication problems.

2. Bugzilla can dramatically increase the productivity and accountability of individual employees by providing a documented workflow and positive feedback for good performance. Ultimately, Bugzilla puts the power in user's hands to improve value to business while providing a usable framework for natural attention to detail and knowledge store to flourish.

The bugzilla utility basically allows to do the following:

- Add a bug into the database
- Review existing bug reports
- Manage the content

Bugzilla is organized in the form of bug reports that give all the information needed about a particular bug. A bug report would consist of the following fields.

- Product–>Component
- Assigned to
- Status (New, Assigned, Fixed etc)

- Summary
- Bug priority
- Bug severity (blocker, trivial etc)
- Bug reporter

Using Bugzilla:

Bugzilla usage involves the following activities

- Setting Parameters and Default Preferences
- Creating a New User
- Impersonating a User
- Adding Products
- Adding Product Components
- Modifying Default Field Values
- Creating a New Bug
- Viewing Bug Reports

Setting Parameters and Default Preferences:

When we start using Bugzilla, we'll need to set a small number of parameters and preferences. At a minimum, we should change the following items, to suit our particular need:

▪ Set the *maintainer*

▪ Set the *mail_delivery_method*

▪ Set *bug change policies*

▪ Set the display order of bug reports

To set parameters and default preferences:

1. Click *Parameters* at the bottom of the page.
2. Under *Required Settings*, add an email address in the *maintainer* field.
3. Click *Save Changes*.
4. In the left side *Index* list, click *Email*.
5. Select from the list of mail transports to match the transport we're using. If evaluating a click2try application, select *Test*. If using SMTP, set any of the other SMTP options for your environment. Click *Save Changes*.
6. In the left side *Index* list, click *Bug Change Policies*.
7. Select On for *comment on create*, which will force anyone who enters a new bug to enter a comment, to describe the bug. Click *Save Changes*.
8. Click *Default Preferences* at the bottom of the page.

9. Select the display order from the drop-down list next to the *When viewing a bug, show comments in this order* field. Click *Submit Changes*.

Creating a New User

Before entering bugs, make sure we add some new users. We can enter users very easily, with a minimum of information. Bugzilla uses the email address as the user ID, because users are frequently notified when a bug is entered, either because they entered the bug, because the bug is assigned to them, or because they've chosen to track bugs in a certain project.

To create a new user:

1. Click **Users**.
2. Click add a new user.
3. Enter the **Login name**, in the form of an email address.
4. Enter the **Real name**, a password, and then click **Add**.
5. Select the **Group access options**. we'll probably want to enable the following options in the row titled User is a member of these groups:
   - *canconfirm*
   - *editbugs*
   - *editcomponents*
6. Click **Update** when done with setting options.

Impersonating a User

Impersonating a user is possible, though rare, that we may need to file or manage a bug in an area that is the responsibility of another user when that user is not available. Perhaps the user is on vacation, or is temporarily assigned to another project. We can impersonate the user to create or manage bugs that belong to that user.

Adding Products

We'll add a product in Bugzilla for every product we are developing. To start with, when we first login to Bugzilla, we'll find a test product called **TestProduct**. We should delete this and create a new product.

To add a product:

1. At the bottom of the page, click **Products**.
2. In the **TestProduct** listing, click **Delete**.
3. Click **Yes, Delete**.
4. Now click **Add a product**.

5. Enter a product name, such as "Widget Design Kit."
6. Enter a description.
7. Click **Add**. A message appears that you'll need to add at least one component.

Adding Product Components

Products are comprised of components. Software products, in particular, are typically made up of many functional components, which in turn are made up of program elements, like classes and functions. It's not unusual in a software development team environment for different individuals to be responsible for the bugs that are reported against a given component. Even if there are other programmers working on that component, it's not uncommon for one person, either a project lead or manager, to be the gatekeeper for bugs. Often, they will review the bugs as they are reported, in order to redirect them to the appropriate developer or even another team, to review the priority and severity supplied by the reporter, and sometimes to reject bugs as duplicates or enhancement requests, for example.

To add a component:

1. Click the link **add at least one component** in the message that appears after creating a new product.
2. Enter the **Component** name.
3. Enter a **Description**.
4. Enter a **default assignee**. Use one of the users we've created. Remember to enter the assignee in the form of an email address.
5. Click **Add**.
6. To add more components, click the name of product in the message that reads edit other components of product <**product name**>.

Modifying Default Field Values

Once we begin to enter new bugs, we'll see a number of drop-down lists containing default values. Some of these may work just fine for our product. Others may not. We can modify the values of these fields, adding new values and deleting old ones. Let's take a look at the OS category.

To modify

To modify default field values:

1. At the bottom of the page, in the **Edit** section, click **Field Values**.
2. Click the link, in this case **OS**, for the field we want to edit. The OS field contains a list of operating system names. We are going to add browsers to this list. In reality, we might create a custom field instead, but for the sake of this example, just add them to the OS list.
3. Click **Add a value**. In the **Value** field, enter "IE7." Click **Add**.
4. Click **Add a value** again.

5. In the **Value** field, enter "Firefox 3."
6. Click **Add**.
7. Where it reads **Add other values for the op_sys field**, click **op_sys**.
8. This redisplays the table. We should now see the two new entries at the top of the table. These values will also appear in the OS drop-down list when we create a new bug.

Creating a New Bug

Creating bugs is a big part of what Bugzilla does best.

To create a new bug:

1. In the top menu, click **New**.
2. If we've defined more than one component, choose the component from the component list.
3. Select a **Severity** and a **Priority**. **Severity** is self-explanatory, but **Priority** is generally assumed to be the lower the number, the higher the priority. So, a **P1** is the highest priority bug, a *showstopper*.
4. Click the **OS** drop-down list to see the options, including the new browser names we entered.
5. Select one of the options.
6. Enter a summary and a description. We can add any other information of choice, but it is not required by the system, although we may determine that our bug reporting policy requires certain information.
7. Click **Commit**. Bugzilla adds our bug report to the database and displays the detail page for that bug.

Viewing Bug Reports

Eventually, we'll end up with thousands of bugs listed in the system. There are several ways to view the bugs. The easiest is to click the My Bugs link at the bottom of the page. Because we've only got one bug reported, we'll use the standard Search function.

To find a bug:

1. Click **Reports**.
2. Click the **Search** link on the page, not the one in the top menu. This opens a page titled "Find a Specific Bug."
3. Select the **Status**.
4. Select the **Product**.
5. Enter a word that might be in the title of the bug.

6. Click **Search**. If any bugs meet the criteria that we have entered, Bugzilla displays them in a list summary.
7. Click the **ID** number link to view the full bug report.

Modifying Bug Reports

Suppose we want to change the status of the bug. We've reviewed it and have determined that it belongs to one of the users we have created earlier

To modify a bug report:

1. Scroll down the full bug description and enter a comment in the **Additional Comments** field.
2. Select "Reassign bug to" and replace the default user ID with one of the other user IDs you created. It must be in the format of an email address.

Experiment: Study of Any Test Management Tool (Test Director)

Test Director is a global test management solution which provides communication, organization, documentation and structure to the testing project.

Test Director is used for

- Mapping Requirements to User acceptance test cases
- Test Planning by placing all the test cases and scripts in it.
- Manual testing by defining test steps and procedures
- Test Execution status
- Defect Management

The Test Director Testing Process

Test Director offers an organized framework for testing applications before they are deployed. Since test plans evolve with new or modified application requirements, you need a central data repository for organizing and managing the testing process. Test Director guides through the requirements specification, test planning, test execution, and defect tracking phases of the testing process.

The Test Director testing process includes four phases:

Specifying Requirements

- Requirements are linked to tests and defects to provide complete traceability and aid the decision-making process
- See what percent of requirements are covered by tests
- Each requirement in the tree is described in detail, and can include any relevant attachments. The QA tester assigns the requirement a priority level which is taken into consideration when the test team creates the test plan
- Import from Microsoft Word or third party RM tool

Planning Tests

- The Test Plan Manager enables to divide application according to functionality. Application can be divided into units, or subjects, by creating a test plan tree.
- Define subjects according to:
    - Application functionality-such as editing, file operations, and reporting
    - Type of testing-such as functional, user interface, performance, and load
- As the tests are also linked to defects, this helps ensure compliance with testing requirements throughout the testing process

Running Tests

- As the application constantly changes, using test lab, run manual and automated tests in the project in order to locate defects and assess quality.
- By creating test sets and choosing which tests to include in each set, test suite can be created? A test set is a group of tests in a Test Director project database designed to achieve specific testing goals.
- Tests can be run manually or scheduled to run automatically based on application dependencies.

Tracking Defects

Locating and repairing application defects efficiently is essential to the testing process. Defects can be detected and added during all stages of the testing process. In this phase you perform the following tasks:

- This tool features a sophisticated mechanism for tracking software defects, enabling Testing Team and the project Team to monitor defects closely from initial detection until resolution
- By linking Test Director to e-mail system, defect tracking information can be shared by all Development and Management Teams, Testing and Wipro Software Quality Assurance personnel

System Requirements for Test Director

Server System configuration : 128 MB of RAM , 500 MB of free disk space, Win NT server, Win 2K server, IIS 5.0, MS-Access/Oracle 7.x,8.x,9/MS SQL Server

Client System configuration : 64 MB of RAM , 10 MB of free disk space, Win 95/98/NT/2K/XP, IE 5 , Netscape 4.7

Experiment: Study of any open source testing tool (Test Link)

Test link is an open source test management tool. It enables creation and organization of test cases and helps manage into test plan. It allows execution of test cases from test link itself. One can easily track test results dynamically, generate reports, generate test metrics, prioritize test cases and assign unfinished tasks.

It's a web based tool with GUI, which provides an ease to develop test cases, organize test cases into test plans, execute these test cases and generate reports.

Test link exposes API, written in PHP, can help generate quality assurance dashboards. The functions like AddTestCaseToTestPlan, Assign Requirements, Create TestCase etc. helps create and organize test cases per test plan. Functions like GetTestCasesForTestPlan, GetLastExecutionResult allows one to create quality assurance dashboard.

TestLink enables easily to create and manage Test cases as well as organize them into Test plans. These Test plans allow team members to execute Test cases and track test results dynamically, generate reports, trace software requirements, prioritize and assign tasks. Read more about implemented features and try demo pages.

Overall structure

There are three cornerstones: **Product**, **Test Plan** and **User**. All other data are relations or attributes for this base. First, define a couple of terms that are used throughout the documentation.

Products and Test Plans

Product: A Product is something that will exist forever in Test Link. Products will undergo many different versions throughout their life times. Product includes Test Specification with Test Cases and should be sorted via Keywords.

Test Plan: Test Plans are created when you'd like to execute test cases. Test plans can be made up of the test cases of one or many Products. Test Plan includes Builds, Test Case Suite and Test Results.

User: A User has a Role that defines available Test Link features.

Test Case Categorization

Test Link breaks down the test case structure into three levels:

Component: Components are parents of Categories. Component can have many Categories.

Category: Categories are the parents of test cases. Each Category can have many test cases.

Test Case: Test cases are the fundamental piece of Test Link.

Test Specification

Creating Test Cases

Tester must follow this structure: Component, Category and test case. At first you create Component(s) for your Product. Component includes Categories. Category has the similar meaning but is second level of Test Specification and includes just Test Cases.

User can also copy or move Test Cases.

Test Cases have following parts:

• Title: could include either short description or abbreviation (e.g. TL-USER-LOGIN)

• Summary: should be really short; just for overview.

• Steps: describe input actions; can also include precondition and cleanup information here.

• Expected results: describe expected behavior of a tested Product or system.

Deleting Test Cases

Test cases, Categories, and Components may be deleted from a test plan by users with lead permissions from the "delete test cases" screen. Deleting data may be useful when first creating a test plan since there are no results. However, Deleting test cases will cause the loss of all results associated with them. Therefore, extreme caution is recommended when using this functionality.

Requirements relation

Test cases could be related with software/system requirements as n to n. The functionality must be enabled for a Product. User can assign Test Cases and Requirements via link Assign Requirements in the main screen.

Test Plans

Test plan contains name, description, collection a chosen test cases, builds, test results, milestones, tester assignment and priority definition.

Creating a new Test Plan

Test Plans may be deleted from the “Create test plan” page (link “Create Test Plan”) by users with lead privileges. Test plans are the basis for test case execution. Test plans are made up of test cases imported from Products at a specific point of time. Test plans can only be created by users with lead privileges. Test plans may be created from other test plans. This allows users to create test plans from test cases that at a desired point in time. This may be necessary when creating a test plan for a patch. In order for a user to see a test plan they must have the proper rights. Rights may be assigned (by leads) in the define User/Project Rights section. This is an important thing to remember when users tell you they can't see the project they are working on.

Test Execution

Test execution is available when:

1. A Test Specification is written.

2. A Test Plan is created.

3. Test Case Suite (for the Test Plan) is defined.

4. A Build is created.

5. The Test plan is assigned to testers (otherwise they cannot navigate to this Test Plan).

Select a required Test Plan in main page and navigate to the 'Execute tests' link. Left pane serves for navigation in Test Case Suite via tree menu, filtering and define a tested build.

Test Status

Execution is the process of assigning a result (pass, fail, blocked) to a test case for a specific build. 'Blocked' test case is not possible to test for some reason (e.g. a problem in configuration disallows to run a tested functionality).

Insert Test results

Test Results screen is shown via click on an appropriate Component, Category or test case in navigation pane. The title shows the current build and owner. The colored bar indicate status of the test case. Yellow box includes test scenario of the test case.

Updated Test Cases

If users have the proper rights they can go to the "Update modified test case" page through the link on main page. It is not necessary for users to update test cases if there has been a change (newer version or deleted).

Advantages:

1. Easy in tracking test cases (search with keyword, test case id, version etc)
2. We can add our custom fields to test cases.
3. Allocating the work either test case creation/execution any kind of documents is easy
4. When a test case is updated the previous version also can be tracked
5. Test plans are created for builds and work allocations can be done.
6. Report is one of the awesome functionality present in the Test link, it generates reports in desired format like HTML/ CSV /Excel and we can create graphs too.
7. And the above all is done on the privileges based which is an art of the test link and i liked this feature much

Example of Test Link workflow:

1. Administrator creates a Product "Fast Food" and a user Adam with rights "leader" and Bela with rights "senior tester".

2. Adam imports Software Requirements and for part of these requirements generates empty Test cases.

3. Bela describes test scenario of these Test cases that are organized according to Components and Categories.

4. Adam creates Keyword: "Regression" and assigns this keyword to ten of these test cases.

5. Adam creates a Test Plan "Fish & Chips", Build "Fish 0.1" and adds Test Cases with keywords "Regression".

6. Adam and Bela execute and record the testing with result: 5 passed, 1 failed and 4 is blocked.

7. Developers make a new build "Fish 0.2" and Bela tests the failed and blocked test cases only. Exceptionaly all these five Test cases passed.

8. Manager would like to see results. Administrator explains him that he can create account himself on the login page. Managers do it. He has "Guest" rights and could see results and Test cases. He can see that everything passed in overall report ;-) and problems in build "Fish 0.1" in a report for particular build. But he can change.

VIVA QUESTIONS:

1. What is UML?

UML is Unified Modeling Language. It is a graphical language for visualizing specifying constructing and documenting the artifacts of the system. It allows you to create a blue print of all the aspects of the system, before actually physically implementing the system.

2. What is modeling? What are the advantages of creating a model?

Modeling is a proven and well-accepted engineering technique which helps build a model. Model is a simplification of reality; it is a blueprint of the actual system that needs to be built. Model helps to visualize the system. Model helps to specify the structural and behavior of the system. Model helps make templates for constructing the system. Model helps document the system.

3. What are the different views that are considered when building an object-oriented software system?

Normally there are 5 views. Use Case view - This view exposes the requirements of a system. Design View - Capturing the vocabulary. Process View - modeling the distribution of the systems processes and threads. Implementation view - addressing the physical implementation of the system. Deployment view - focus on the modeling the components required for deploying the system.

4. What are diagrams?

Diagrams are graphical representation of a set of elements most often shown made of things and associations.

5. What are the major three types of modeling used?

Major three types of modeling are structural, behavioral, and architectural.

6. Mention the different kinds of modeling diagrams used?

Modeling diagrams that are commonly used are, there are 9 of them. Use case diagram, Class Diagram, Object Diagram, Sequence Diagram, state chart Diagram, Collaboration Diagram, Activity Diagram, Component diagram, Deployment Diagram.

7. What is Architecture?

Architecture is not only taking care of the structural and behavioral aspect of a software system but also taking into account the software usage, functionality, performance, reuse, economic and technology constraints.

8. What is SDLC?

SDLC is Software Development Life Cycle. SDLC of a system included processes that are Use case driven, Architecture centric and Iterative and Incremental. This Life cycle is divided into phases. Phase is a time span between two milestones. The milestones are Inception, Elaboration, Construction, and Transition. Process Workflows that evolve through these phase are Business Modeling, Requirement gathering, Analysis and Design, Implementation, Testing, Deployment. Supporting Workflows are Configuration and change management, Project management.

9. What are Relationships?

There are different kinds of relationships: Dependencies, Generalization, and Association. Dependencies are relations ships between two entities that that a change in specification of one thing may affect another thing. Most commonly it is used to show that one class uses another class as an argument in the signature of the operation. Generalization is relationships specified in the class subclass scenario, it is shown when one entity inherits from other. Associations are structural relationships that are: a room has

walls, Person works for a company. Aggregation is a type of association where there is a has a relationship, That is a room has walls, if there are two classes room and walls then the relationship is called a association and further defined as an aggregation.

10. How are the diagrams divided?

The nine diagrams are divided into static diagrams and dynamic diagrams.

Static Diagrams (Also called Structural Diagram)

Class diagram, Object diagram, Component Diagram, Deployment diagram.

Dynamic Diagrams (Also called Behavioral Diagrams)

Use Case Diagram, Sequence Diagram, Collaboration Diagram, Activity diagram, State chart diagram.

11. What are Messages?

A message is the specification of a communication, when a message is passed that results in action that is in turn an executable statement.

12. What is a Use Case?

A use case specifies the behavior of a system or a part of a system, Ã³se cases are used to capture the behavior that need to be developed. It involves the interaction of actors and the system.

13. Define Object Oriented Analysis?

Object Oriented Analysis (OOA) is a method of analysis that examines requirements from the perspective of the classes and objects found in the vocabulary of the problem domain.

14. What is meant by Object Oriented?

Object Oriented means we organize the software as a collection of discrete objects that incorporate both data structure and behavior.

15. Write the characteristics of an object.

Identity, classification, polymorphism, and inheritance.

16. What is a class?

A class is a set of objects that share a common structure and a common behavior.

17. Name two types of object diagram.

Class diagram and instance diagram.

18. What is an attribute? Give example.

An attribute is a data value held by the objects in a class .Example: name, age and weight are attributes of Person class.

19. What is multiple inheritances?

When one class inherits its state (attributes) and behavior from more than one super class, it is referred to as multiple inheritances.

20. What is dynamic binding?

The process of determining (dynamically) at run time which functions to invoke is termed dynamic binding.

21. What is static binding?

The process of determining at compile time which functions to invoke is termed static binding.

22. Write the four quality measures for software development?

Correspondence, correctness, verification, and validation.

23. What is object persistence?

Objects have life time. They are created and can exist for a period of time.

A file or a database can provide support for objects having a longer life time longer than the duration of the process for which they were created. This characteristic is called object persistence.

24. What is polymorphism? Give an example.

Polymorphism means that the same operation may behave differently on different classes.

Ex: Move operation. (Behave differently on the window class and chess Piece class).

25. What is cardinality?

Cardinality specifies how many instances of one class may relate to a single instance of an associated class.

26. What is a formal class or abstract class?

Formal or abstract classes have no instances but define the common behaviors that can be inherited by more specific classes.

27. What is a meta-class?

A meta-class is a class about a class. They are normally used to provide instance variables and operations.

28. Define Encapsulation?

Encapsulation is the process of compartmentalizing the elements of an abstraction that constitute its structure and behavior.

29. What is the need of an Object diagram?

An object diagram is used to show the existence of objects and their relationships in the logical design of a system.

30. What is state of an object?

The state of an object encompasses all of the properties of the object plus the current values of each of these properties.

31. Write some applications of object model?

They include Air traffic control, Animation, Avionics, Database, Robotics etc.

32. Define Concurrency.

Concurrency is the property that distinguishes an active object from one that is not active.

33. Name the three general approaches for classification?

They are Classical categorization, Conceptual clustering and Prototype theory.

34. Name the five levels of process maturity in OOD?

They are Initial, Repeatable, Defined, Managed and Optimized.

35. Name the two process used by Grady BOOCH in his OO software development?

They are Macro and Micro development process.

36. Name the four steps in Micro development process?

They are Identify the classes and objects, Give semantics to the classes, Identify class and object relationships, Identify class and object interfaces and implementation.

37. What are the steps followed in macro development process?

Conceptualization, analysis and development of the model, Design or create the system architecture, evolution or implementation, maintenance.

38. Short notes on OMT functional model?

OMT functional model uses dataflow diagram that shows the flow of data between different processes in a business .Data flow diagrams use four primary symbols. They are process, data flow, data store, external entity.

39. Names the diagrams of Booch Methodology?

Class diagram, object diagram, state transition diagram, use case diagram, collaboration diagram, sequence diagram , interaction diagram, component diagram, deployment diagram.

40. Name the models in objectory.

Use case model, domain object model, analysis object model, implementation model, test model.

41. What is unified modeling language?

Unified modeling language is a language for specifying, conducting, visualizing and documenting the software system and its components.

42. Name the available layers of the three layered approach to software development.

Business layer, access layer, view (user interface) layer.

43. Write the two responsibilities of access layer?

Translate Request, Translate result.

44. Write any two advantages of modeling?

The main reason for modeling is the reduction of complexity. The cost of the modeling analysis is much lower than the cost of similar experimentation conducted with real time.

45. What is Objectory?

Objectory, is a method or object-oriented development with the specific aim to fit the development of large, real-time systems

46. Define Static model?

It can be viewed as a snapshot of a system's parameters at rest or a specific point in time. They are needed to represent the structural or static aspect of a system.

47. Define Dynamic model?

It can be viewed as a collection of procedures or behaviors that taken together reflect the behavior of a system over time. Dynamic modeling is the most useful during the design and implementation phases of the system development.

48. What is an association? Give one example.

An association is the relationship between the classes.

Ex person and company are the classes, works-for is the association name.

49. What is a qualifier? Give one example.

A qualifier is an association attribute. The qualifier rectangle is part of the association path, not part of the class.

50. What is a method?

A method is the implementation of an operation for a class.

51. What is a use case?

Use cases are scenarios for understanding system requirements. A use case is an interaction between users and a system.

52. Name the three types of relationships in a use case diagram.

Communication, Uses, extends.

53. Write the two types of Implementation diagram?

Component diagram, deployment diagram.

54. What is an activity?

An activity is a set of operations that is executing during the entire period an object is in a state.

55. Write the guidelines for preparing the Documentation.

Common cover, 80-20 rule, Familiar vocabulary, make the document as short as possible, organize the document.

56. Name the types of relationships among the objects.

Association, super-sub structure, aggregation.

57. Write the guidelines for identifying the associations

A dependency between two or more classes may be association. A reference from one class to another is an association.

58. Name the two properties of a part of relationship.

Transitivity, Anti symmetry.

59. Write the Guidelines for identifying part of relationship.

Assembly, container, collection member

60. Define Prototype?

A prototype is a version of a software product developed in the early stages of the product's life cycle for specific, experimental purposes. A prototype enables you to fully understand how easy or difficult it will be to implement some of the features of the system.

61. Define pattern mining?

The process of looking for patterns to document is called pattern mining Some times called reverse architecture.

62. Define anti-patterns?

An anti-pattern represents a worst practice while a pattern represents a best Practice. Anti-patterns come in two varieties. Those describing a bad solution to a problem that resulted in a bad situation and Those describing how to get out of a bad situation.

63. Define patterns template?

Every pattern must be expressed in the form of a rule which is called as a Template. It should establish a relationship between a context, a system of forces which arises in the context, and a configuration.

64. Define proto-patterns?

If something appears to have all the requisite pattern components, it should not be considered a pattern until it has been verified to be a recurring phenomenon .A proto-pattern is the "pattern in waiting" which is not yet known to recur.

65. Name the two categories of Quality assurance testing.

Error based testing, scenario based testing.

66. Define debugging.

Debugging is the process of finding out where something went wrong and correcting the code to eliminate the errors or bugs that cause unexpected results.

67. Write the two types of path testing.

Statement testing coverage and Branch testing coverage.

68. What is a meta-model?

A meta-model is a model of modeling elements. UML graphic notations can be used not only to describe the system's components but also to describe a model itself.

69. Define a Framework?

A frame work is a collection of classes that provide a set of services for a particular domain.

70. Write the differences between design patterns and frameworks

- Design patterns are more abstract than frameworks.
- Design patterns are smaller architectural elements than frameworks.
- Design patterns are less specialized than frameworks.

71. Define SQA?

SQA stands for Software Quality Assurance. This is the measure of assuring the quality of the software products. The major activity done here is testing. The assurance process also follows the quality model called the QAIMODEL (Quality Assurance Institute Model).

72. What is V Testing?

'V' testing stands for Verification and Validation testing.

73. What is a quality?

Quality refers to the ability of products to meet the user's needs and expectations.

74. Name the two issues for software quality.

Validation or user satisfaction, and verification or quality assurance.

75. Define user satisfaction testing.

User satisfaction testing is the process of quantifying the usability test with some measurable attributes of the test, such as functionality, cost or ease of use.

76. Define test plan.

A test plan is developed to detect and identify potential problems before delivering the software to its users.

77. Write the objectives of testing.

Testing is the process of executing a program with the intent of finding errors.

A good test case is the one that has a high probability of detecting an as yet undiscovered error.

A successful test case is the one that detects an as yet undiscovered error.

78. What is cyclomatic complexity?

Cyclomatic complexity is software metric that provides a quantitative measure of the logical complexity of a program. The value computed for cyclomatic complexity defines the number of independent paths in the basis set of program.

79. Define corollary?

Corollary is a proposition that follows from an axiom or another proposition that has been proven.

Name the two axioms.

Axiom1: The independence axiom. Maintain the independence of components.

Axiom2: The information axiom. Minimize the information content of the design.

80. Define coupling.

Coupling is a measure of the strength of association established by a connection from one object or software component to another. Coupling is a binary relationship. Coupling deals with interactions between objects or software components.

81. Name the two types of coupling in the object oriented design.

Interaction coupling and inheritance coupling.

82. Define cohesion.

Cohesion means the interactions within a single object or software component.

83. Name the types of attributes.

Single value attribute, Multiplicity or multi-value attributes, Reference to another object or instance connection.

84. Write the syntax for presenting the attribute that was suggested by UML.

visibility name : type_expression = initial _value

Where visibility is one of the following

+ public visibility

protected visibility

- private visibility

type_expression - type of an attribute

Initial_value is a language dependent expression for the initial value of a newly created object.

85.Write the syntax for presenting the operation that was suggested by UML

Visibility name: (parameter_list): return _type_expression

Where visibility is one of the following

+ public visibility

protected visibility

- private visibility

parameter- is a list of parameters.

Return_type_expression: is a language _dependent specification of the

Implementation of the value returned by the method.

86. What is a Façade?

Facade classes are the public classes in a package for public behavior.

87. Define DBMS?

A database management system (DBMS) is a program that enables the creation and maintenance of a collection of related data.

88. What is database model?

Database model is a collection of logical constructs used to represent the data structure and data relationships within the database.

89. Name the two categories of database model?

Conceptual model and Implementation model.

90. Write the six categories for the life time of data

Transient results to the evaluation of expressions, variables involves in procedure activation, global variables and variables that are dynamically allocated, data that exist between the execution of a program, data that exist between the versions of a program, data that outlive a program.

91. What is schema or metadata?

The fundamental characteristic of the database is that the DBMS contains not only the data but the complete definition of the data formats such as data structures, types and constraints, it manages. This description is known as the schema or metadata

92. Name the three types of data base model?

Hierarchical model, network model, relational model.

93. Define data definition language.

Data definition language (DDL) is a language used to describe the structure of and relationships between objects stored in a database .This structure of information are termed as database schema.

94. Define data manipulation language.

Data manipulation language (DML) is a language that allows users to access and manipulate (such as create, save, or destroy) data organization.

95. When the transaction is said to commit.

The transaction is said to commit if all changes can be made successfully to the database.

96. When the transaction is said to abort.

The transaction is said to abort if all changes to the database can not be made successfully.

97. What is conservative or pessimistic policy?

The most conservative way to enforce serialization is to allow a user to lock all objects or records when they are accessed and to release the locks only after a transaction commits. This approach is known as conservative or pessimistic policy.

98. Describe client server computing.

The client is a process (program) that sends a message to a server process (program) requesting that the server perform a task (service).

99. Name the types of object relation mapping.

Table class mapping, Table –multiple classes mapping, Table-Inherited classes mapping, Tables-Inherited classes mapping.

100. Write the need of middleware.

The client is a process (program) that sends a message to a server process (program) requesting that the server perform a task (service). The key element of connectivity is the network operating system (NOS), also known as middleware.

101. Mention the different forms of server.

File server, database server, transaction server, application server.

102. What is the use of application web server?

In a two-tier architecture, a client talks directly to a server, no intervening server. Three_ tier architecture introduces a server that is application web server between the client and the server to send and receive the messages.

103. Write the components of client server application.

User interface, business processing, database processing.

104. What is Object Oriented Database management system?

Object Oriented Database management system is a marriage of Object Oriented programming and Database management system.

105. Define ODBC?

The Open Database connectivity is an application programming interface that provides solutions to the multi database programming interface.

106. What is the need of an Interaction diagram?

An Interaction diagram is used to trace the exception of a scenario in the same context of an object diagram.

107. What is the need of a Class diagram?

A class diagram is used to show the existence of classes and their relationships in the logical view of a system.

108. What is Behavior of an object?

Behavior is how an object acts and reacts in terms of its state changes and message passing.

109. What are the characteristic features of an Interaction diagram?

They include the representation of objects with its name and class name. Each object has a life line. The order of messaging between objects is well defined.

110. Define forward engineering and revere engineering.

Forward engineering means creating a relational schema from an existing object model

Reverse engineering means creating an object model from an existing relational database layout (schema).

111. What is Object request broker (ORB)?

Object request broker (ORB) –Middle ware that implements a communication channel through which applications can access object interfaces and request data and services.

112. What is distributed database?

In distributed database, different portions of the database reside on different nodes (computers) and disk drives in the network. Each portions of the database is managed by a server, a process responsible for controlling access and retrieval of data from the database portion.

113. What does RAD stands for?

Rapid application development (RAD) is a set of tools and techniques that can be used to build an application faster than typically possible with traditional methods.

114. What are the traditional software development methodologies?

Most traditional development methodologies are either algorithm centric or data centric.

115. What is the MAIN benefit of designing tests early in the life cycle?

It helps prevent defects from being introduced into the code.

116. What is risk-based testing?

Risk-based testing is the term used for an approach to creating a test strategy that is based on prioritizing tests by risk. The basis of the approach is a detailed risk analysis and prioritizing of risks by risk level. Tests to address each risk are then specified, starting with the highest risk first.

117. A wholesaler sells printer cartridges. The minimum order quantity is 5. There is a 20% discount for orders of 100 or more printer cartridges. You have been asked to prepare test cases

using various values for the number of printer cartridges ordered. Which of the following groups contain three test inputs that would be generated using Boundary Value Analysis?

4, 5, 99

118. What is the KEY difference between preventative and reactive approaches to testing?

Preventative tests are designed early; reactive tests are designed after the software has been produced.

119. What is the purpose of exit criteria?

To define when a test level is complete.

120. What determines the level of risk?

The likelihood of an adverse event and the impact of the event

121. When is used Decision table testing?

Decision table testing is used for testing systems for which the specification takes the form of rules or cause-effect combinations. In a decision table the inputs are listed in a column, with the outputs in the same column but below the inputs. The remainder of the table explores combinations of inputs to define the outputs produced.

Learn More about Decision Table Testing Technique in the Video Tutorial here

122. What is the MAIN objective when reviewing a software deliverable?

To identify defects in any software work product.

123. Which of the following defines the expected results of a test? Test case specification or test design specification.

Test case specification.

124. Which is a benefit of test independence?

It avoids author bias in defining effective tests.

125. As part of which test process do you determine the exit criteria?

Test planning.

126. What is beta testing?

Testing performed by potential customers at their own locations.

REFERENCES:

CASE TOOLS

1. J.A.Hoffer, J.F.George and J.S.Valacich "Modern Systems Analysis and Design", Pearson Education Asia, New Delhi, 2002

Chapter 4 "Automated Tools for Systems Development" has a good discussion of CASE tools.

2. G.Booch, J.Rumbaugh, I.Jacobson; "The Unified Modeling

Language User Guide", Addison Wesley, Reading, MA, USA, KGG authentic Introduction to Rational Commercial tools for Object oriented modeling

3. URL's of various available CASE tools

(a) System flowchart and Er-diagram generation tool: Smartdraw

http://www.smartdraw.com

(b) Data flow diagram tool: IBMS/DFD

http://viu.eng.rpi.edu

(c) Tool to convert decision table to structured english: COPE

http://www.cs.adelaide.edeu.au/users/dwyer/examples.html

SOFTWARE TESTING

1. Software Testing Techniques- Boris Bezier

2. Software Testing Tools – Dr. K.V.K.K. Prasad

3. The craft of software testing -Brian Marick

4. Software Testing Techniques – SPD (Oreille)

5. Software testing in the Real - Edward kit