**Experiment No: 02**

**⚫Aim: Application of the agile process model.**

**⚫Theory:**

**⚫ Project Name: - The QR CODE SCANNER**

**Agile Model**

The meaning of Agile is swift or versatile. **“Agile process model**" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.

Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.



## **Phases of Agile Model:**

Following are the phases in the agile model are as follows:

1. Requirements gathering
2. Design the requirements
3. Construction/ iteration
4. Testing/ Quality assurance
5. Deployment
6. Feedback

**Requirements gathering:** In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.

An App Store QR Code is a QR Code solution that connects users directly to the app store to promote app installations from the Apple Store, Google Play, and Amazon Appstore. Set within a mobile-friendly and customizable page, you can include buttons to each app store where your app is located to streamline the app installation process for customers with all types of devices—all with a single QR Code.

* Android: is a standard GUI and is one of the easiest ways to build a GUI application.
* Java: programing propose
* Firebase: real time database
* Adobe (UI/UX); Build in layout and Design

**Design the requirements:** When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.

**Construction/ iteration:** When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.

**Testing:** In this phase, the Quality Assurance team examines the product's performance and looks for the bug. In testing phase we came to know the types of errors a user might face while using our application. The permission a user needs to provide and the error that might pop up if permission is not provided. The Camera quality is well or not that has been checked while testing process. The QR code is able to scan and decode the URL without fail. Even the Authentication process that stores info regarding user in real time database that is Firebase. We as well checked whether a unique QR is generated every time the QR is been generated.

**Deployment:** In this phase, the team issues a product for the user's work environment. In Deployment stage the application is useful for user and provides security so in existing market this application doesn’t wear out. As nowadays most of sites prefers QR Code to provide that facility this application is beneficial.

**Feedback:** After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback. After Completing the planning part there was still a part missing from it and due to that the whole planning was required to change but the managing team managed whole that part and rebuilt the plan as per required.

**Agile Testing Methods:**

* Scrum
* Crystal
* Dynamic Software Development Method(DSDM)
* Feature Driven Development(FDD)
* Lean Software Development
* eXtreme Programming(XP).

### **Scrum:**

SCRUM is an agile development process focused primarily on ways to manage tasks in team-based development conditions.

There are three roles in it, and their responsibilities are:

1. **Scrum Master:** The scrum can set up the master team, arrange the meeting and remove obstacles for the process
2. **Product owner:** The product owner makes the product backlog, prioritizes the delay and is responsible for the distribution of functionality on each repetition.
3. **Scrum Team:** The team manages its work and organizes the work to complete the sprint or cycle.

## **Advantage of Agile Method:**

1. Frequent Delivery
2. Face-to-Face Communication with clients.
3. Efficient design and fulfils the business requirement.
4. Anytime changes are acceptable.
5. It reduces total development time.

## **Disadvantages of Agile:**

1. Due to the shortage of formal documents, it creates confusion and crucial decisions taken throughout various phases can be misinterpreted at any time by different team members.
2. Due to the lack of proper documentation, once the project completes and the developers allotted to another project, maintenance of the finished project can become a difficulty.

* **Conclusion :-**

We successfully implemented **Application of the agile process model.**

**Experiment No: 04**

**⚫Aim: Structured data flow analysis.**

**⚫Theory:**

**⚫ Project Name: - The QR CODE SCANNER**

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.



**Circle**: A circle (bubble) shows a process that transforms data inputs into data outputs.

**Data** **Flow**: A curved line shows the flow of data into or out of a process or data store

**Data** **Store**: A set of parallel lines shows a place for the collection of data items. A data store indicates that the data is stored which can be used at a later stage or by the other processes in a different order. The data store can have an element or group of elements.

**Source or Sink**: Source or Sink is an external entity and acts as a source of system inputs or sink of system outputs.

Levels in Data Flow Diagrams (DFD):

* 0-level DFD,
* 1-level DFD,
* 2-level DFD

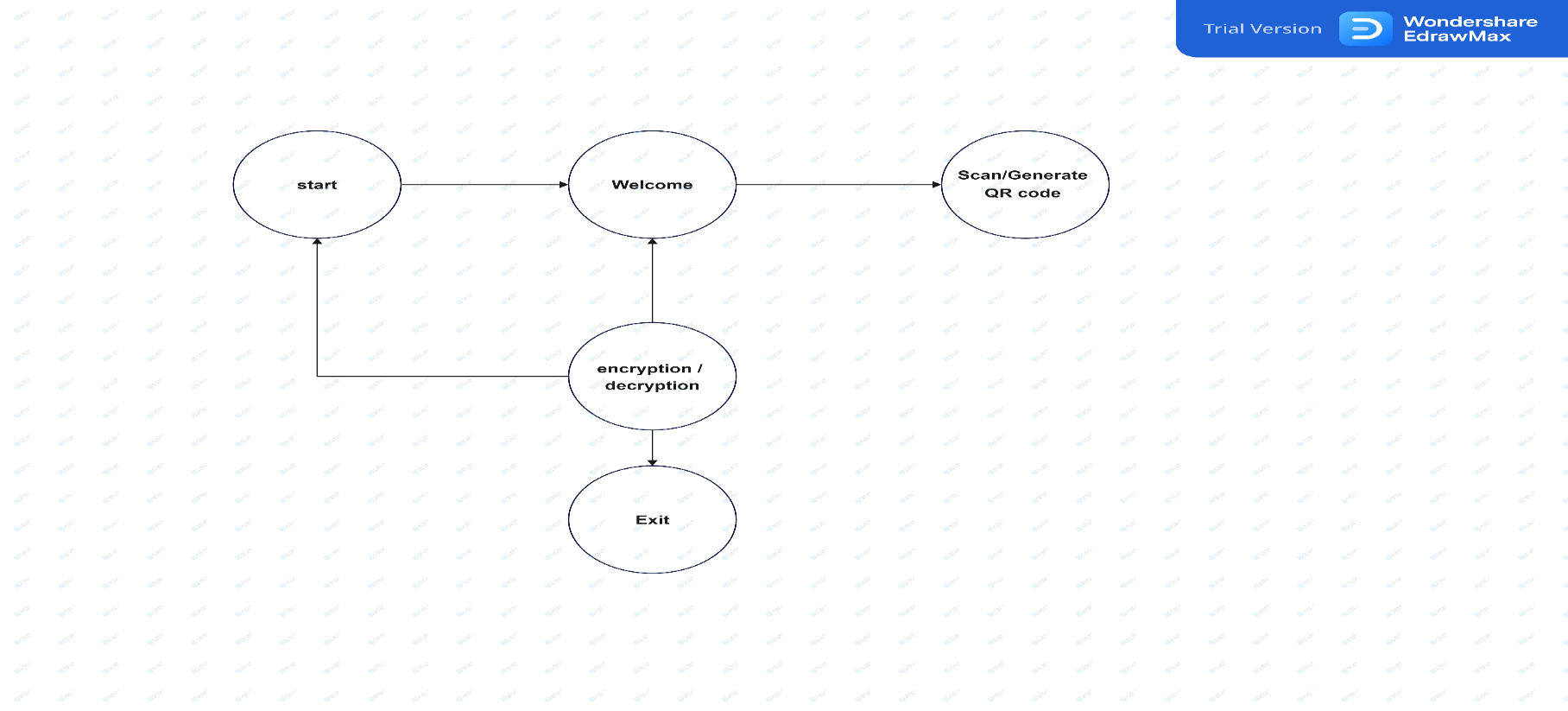
0-level DFD:

**User input User Output**

1-level DFD:

**User input User Output**

2-level DFD:

****

**DFD**

**Decision Trees**

Decision trees are a method for defining complex relationships by describing decisions and avoiding the problems in communication. A decision tree is a diagram that shows alternative actions and conditions within horizontal tree framework. Thus, it depicts which conditions to consider first, second, and so on. Decision trees depict the relationship of each condition and their permissible actions. A square node indicates an action and a circle indicates a condition. It forces analysts to consider the sequence of decisions and identifies the actual decision that must be made.

**Data Dictionary -**

A data dictionary is a structured repository of data elements in the system. It stores the descriptions of all DFD data elements that is, details and definitions of data flows, data stores, and data stored in data stores, and the processes. A data dictionary improves the communication between the analyst and the user. It plays an important role in building a database.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Filed-name | Data-type | Filed-size | Description | Example |
| Email id | text | 20 | It have unique email-id | Collegeinfopk02@gmail.com |
| Password | Number | 10 | It have unique Password | Pass@1234 |
| username | text | 10 | It have username | Priyush02 |
| Mobile No. | Number | 10 | It have unique mobile No | 9511689394 |

**Pseudo-code:**

1 BEGIN:

#TAKE THE LOGIN AUTHICATICATION

IF (AUTHICATICATION== TRUE){

PRINT (“LOGIN SUCCESSFUL”);

} ELSE {

PRINT (“LOGIN UNSUCCESSFUL”)  
}

#TAKE THE USER SCAN

IF (SCAN== ENCRPTION/ DESCRIPTION ){

PRINT (“SCAN ”);

} ELSE {

PRINT (“NOT SCAN ”)

RETURN LOGIN;  
}

#TAKE THE USER GENERATE

IF (SCAN== ENCRPTION/ DESCRIPTION ){

PRINT (“SCAN GENERATE ”);

} ELSE {

PRINT (“NOT SCAN GENERATE ”)

RETURN LOGIN;  
}

EXIT:

**Pseudo-code:**

1. Start application
2. Welcome screen occurs
3. Use the information
4. Scan/Generating QR code.
5. Exit app.

**Advantages** of **DFD**

* It helps us to understand the functioning and the limits of a system.
* It is a graphical representation which is very easy to understand as it helps visualize contents.
* Data Flow Diagram represent detailed and well explained diagram of system components.
* It is used as the part of system documentation file.

**Disadvantages** **of** **DFD**

* At times DFD can confuse the programmers regarding the system.
* Data Flow Diagram takes long time to be generated, and many times due to this reasons analyst are denied permission to work on it.

**⚫ Conclusion: -**

We successfully Understanding **Structured data flow analysis.**

**Experiment No: 05**

**⚫Aim: Use** **of metrics to estimate the cost.**

**⚫Theory:**

**⚫ Project Name: - The QR CODE SCANNER**

**Software Cost Estimation**

For any new software project, it is necessary to know how much it will cost to develop and how much development time will it take. These estimates are needed before development is initiated, but how is this done? Several estimation procedures have been developed and are having the following attributes in common.

* Project scope must be established in advanced.
* Software metrics are used as a support from which evaluation is made.
* The project is broken into small PCs which are estimated individually.
* To achieve true cost & schedule estimate, several options arise.
* Delay estimation
* Used symbol decomposition techniques to generate project cost and schedule estimates.
* Acquire one or more automated estimation tools**.**

**Uses of Cost Estimation**

During the planning stage, one needs to choose how many engineers are required for the project and to develop a schedule.

In monitoring the project's progress, one needs to access whether the project is progressing according to the procedure and takes corrective action, if necessary.

A model may be static or dynamic. In a static model, a single variable is taken as a key element for calculating cost and time. In a dynamic model, all variable are interdependent, and there is no basic variable.

**Software Cost Estimation**

Static, Single Variable Models: When a model makes use of single variables to calculate desired values such as cost, time, efforts, etc. is said to be a single variable model. The most common equation is:

C=aLb

Where C = Costs

L= size

a and b are constants

The Software Engineering Laboratory established a model called SEL model, for estimating its software production. This model is an example of the static, single variable model.

E=1.4L0.93

DOC=30.4L0.90

D=4.6L0.26

Where E= Efforts (Person Per Month)

DOC=Documentation (Number of Pages)

D = Duration (D, in months)

L = Number of Lines per code

Static, Multivariable Models: These models are based on method (1), they depend on several variables describing various aspects of the software development environment. In some model, several variables are needed to describe the software development process, and selected equation combined these variables to give the estimate of time & cost. These models are called multivariable models.

WALSTON and FELIX develop the models at IBM provide the following equation gives a relationship between lines of source code and effort:

E=5.2L0.91

In the same manner duration of development is given by

D=4.1L0.36

The productivity index uses 29 variables which are found to be highly correlated productivity as follows:

Software Cost Estimation

Where Wi is the weight factor for the it variable and Xi={-1,0,+1} the estimator gives Xione of the values -1, 0 or +1 depending on the variable decreases, has no effect or increases the productivity.

**COCOMO Model**

Boehm proposed COCOMO (Constructive Cost Estimation Model) in 1981.COCOMO is one of the most generally used software estimation models in the world. COCOMO predicts the efforts and schedule of a software product based on the size of the software.

The necessary steps in this model are:

* Get an initial estimate of the development effort from evaluation of thousands of delivered lines of source code (KDLOC).
* Determine a set of 15 multiplying factors from various attributes of the project.
* Calculate the effort estimate by multiplying the initial estimate with all the multiplying factors i.e., multiply the values in step1 and step2.

The initial estimate (also called nominal estimate) is determined by an equation of the form used in the static single variable models, using KDLOC as the measure of the size. To determine the initial effort Ei in person-months the equation used is of the type is shown below

**Ei=a\*(KDLOC)b**

**In COCOMO, projects are categorized into three types:**

* **Organic**
* **Semidetached**
* **Embedded**

1.**Organic**: A development project can be treated of the organic type, if the project deals with developing a well-understood application program, the size of the development team is reasonably small, and the team members are experienced in developing similar methods of projects. Examples of this type of projects are simple business systems, simple inventory management systems, and data processing systems.

2. **Semidetached**: A development project can be treated with semidetached type if the development consists of a mixture of experienced and inexperienced staff. Team members may have finite experience in related systems but may be unfamiliar with some aspects of the order being developed. Example of Semidetached system includes developing a new operating system (OS), a Database Management System (DBMS), and complex inventory management system.

3. **Embedded**: A development project is treated to be of an embedded type, if the software being developed is strongly coupled to complex hardware, or if the stringent regulations on the operational method exist. For Example: ATM, Air Trafficcontrol**.**

**Software Cost Estimation:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Item | Unit/hr | Cost/hr | Subtotal | level | % of total |
| **Project Management** |  |  |  | 15000 | 15% |
| Project Manger | 2 | 1500 | 3000 |  |  |
| Project team | 60 | 1000 | 6000 |  |  |
| Contractors |  |  | 500 |  |  |
|  |  |  |  |  |  |
| **Hardware** |  |  |  | 1500 |  |
| Hardware Device | 5 | 150 | 750 |  | 60% |
| Servers | 1 | 10 | 100 |  |  |
|  |  |  |  |  |  |
| **Software** |  |  |  | 3000 | 50% |
| Licensed Software | 1 | 500 | 500 |  |  |
| Software development |  |  | 1000 |  |  |
|  |  |  |  |  |  |
| **Training and Support** |  |  |  | 7000 |  |
| Training cost | 10 | 450 | 4500 |  | 90% |
| Travel cost | 5 | 10 | 500 |  |  |
|  |  |  |  |  |  |
| **Reserves (10%)** |  |  |  | 10000 | 10% |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| **Total Project Cost Estimate** |  |  |  | 50000 |  |

**⚫ Conclusion: -**

We successfully Understanding **of metrics to estimate the cost.**

**Experiment No: 06**

**⚫Aim: Scheduling and Tracking of the Project.**

**⚫Theory:**

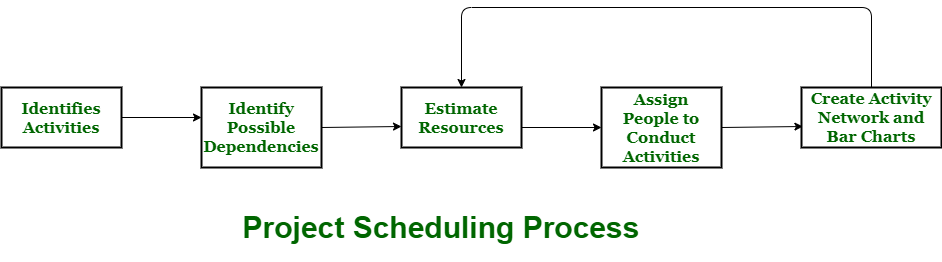
**⚫ Project Name: - The QR CODE SCANNER**

**Project Scheduling:**

Project-task scheduling is a significant project planning activity. It comprises deciding which functions would be taken up when. To schedule the project plan, a software project manager wants to do the following:

* Identify all the functions required to complete the project.
* Break down large functions into small activities.
* Determine the dependency among various activities.
* Establish the most likely size for the time duration required to complete the activities.
* Allocate resources to activities.
* Plan the beginning and ending dates for different activities.
* Determine the critical path. A critical way is the group of activities that decide the duration of the project.

Project scheduling leads to success of project, reduced cost, and increased customer satisfaction. Scheduling in project management means to list out activities, deliverables, and milestones within a project that are delivered. It contains more notes than your average weekly planner notes. The most common and important form of project schedule is Gantt chart.



**Project tracking:**

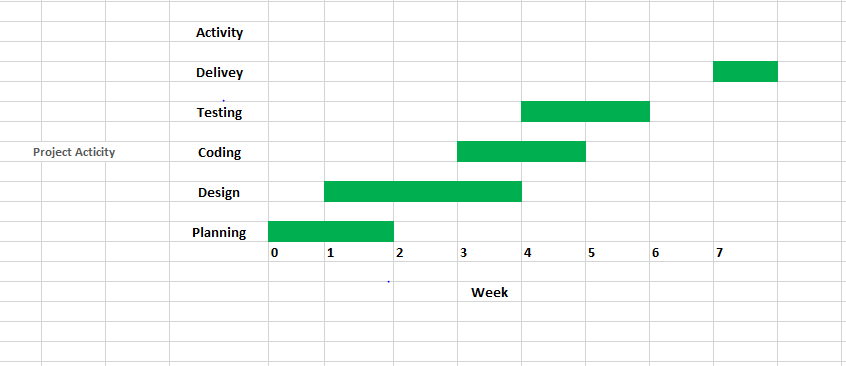
A project tracking system assists you to recognize all of the tasks that are required to complete your project successfully on time. That is why it is recommended to be deployed during planning phase of a project to control and regulate the allocated resources. It provides a fully automated monitoring, and evaluation system of financial requests by simply tracking planned, actual and modified expenditures to information system accounts. The project tracking system provides a highly-standardized, automated technique of budget and planning management across a diverse group of activities.

**Some tools used to schedule and track project**

* Gantt Chart
* PERT Chart
* Resource Histogram
* Critical Path Analysis

**Gantt Chart:**

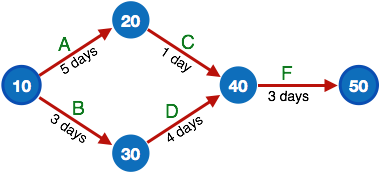
Gantt charts was devised by Henry Gantt (1917). It represents project schedule with respect to time periods. It is a horizontal bar chart with bars representing activities and time scheduled for the project activities.



**Gantt charts**

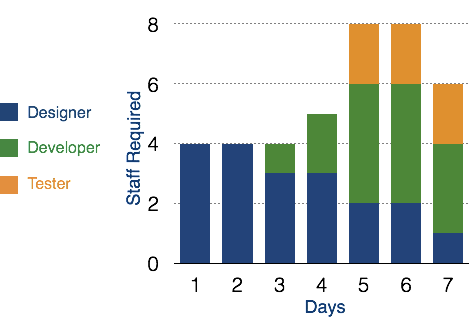
**PERT Chart:**

PERT (Program Evaluation & Review Technique) chart is a tool that depicts project as network diagram. It is capable of graphically representing main events of project in both parallel and consecutive way. Events, which occur one after another, show dependency of the later event over the previous one.



**Resource Histogram:**

This is a graphical tool that contains bar or chart representing number of resources (usually skilled staff) required over time for a project event (or phase). Resource Histogram is an effective tool for staff planning and coordination.



**Critical Path Analysis**

This tool is useful in recognizing interdependent tasks in the project. It also helps to find out the shortest path or critical path to complete the project successfully. Like PERT diagram, each event is allotted a specific time frame. This tool shows dependency of event assuming an event can proceed to next only if the previous one is completed.

The events are arranged according to their earliest possible start time. Path between start and end node is critical path which cannot be further reduced and all events require to be executed in same order.

**Advantages of Project Scheduling:**

There are several advantages provided by project schedule in our project management:

* It simply ensures that everyone remains on same page as far as tasks get completed, dependencies, and deadlines.
* It helps in identifying issues early and concerns such as lack or unavailability of resources.
* It also helps to identify relationships and to monitor process.
* It provides effective budget management and risk mitigation

**⚫ Conclusion: -**

We successfully Understanding **Scheduling and various ways of Tracking the Project.**

**Experiment No: 07**

**⚫Aim: write test cases for black box testing.**

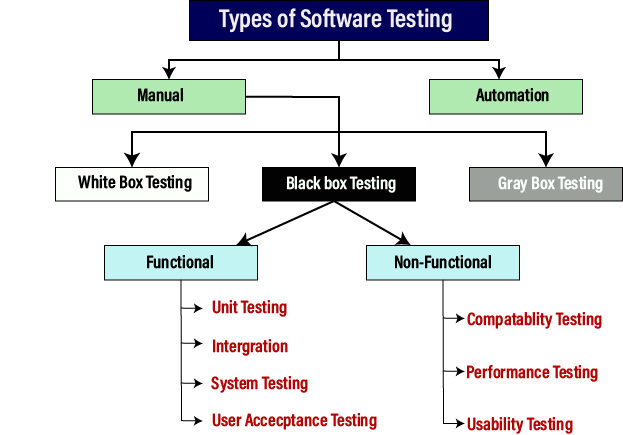
**⚫Theory:**

**⚫ Project Name: - The QR CODE SCANNER**

**Software testing:**

Software testing is a process of identifying the correctness of software by considering its all attributes (Reliability, Scalability, Portability, Re-usability, Usability) and evaluating the execution of software components to find the software bugs or errors or defects.

**Type of Software testing:**



**Black box testing:**

Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioural Testing.



**Types of Black Box Testing:**

There are many types of Black Box Testing but the following are the prominent ones –

* **Functional testing** – This black box testing type is related to the functional requirements of a system; it is done by software testers.
* **Non-functional testing** – This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, usability.
* **Regression testing** – Regression Testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code**.**

**How to do Black Box testing?**

When you get the basic understanding of black-box testing then the next question which

comes up in mind is: How to perform the Black box testing? Below you can check the steps

to perform this testing:

* The first step to black-box testing is to understand the requirement specifications of the application under test. An accurate and precise SRS document should be there.
* The next step is to evaluate the set of valid inputs and test scenarios to test the software.  The goal is to save time and get good test coverage.
* Prepare the test cases to cover a maximum range of inputs.
* The test cases are run in the system to generate output, which is validated with the expected outcome to mark pass or fail.
* The failed steps are marked and sent to the development team to fix them.
* Retest the system using various testing techniques to verify its recurring nature or to pass it.
* The black box testing can be easily used to check and validate the entire software
* development life cycle. It can be used at various stages such as unit, integration, acceptance, system, and regression to evaluate the product.

**Techniques Used in Black Box Testing:**

* Decision Table Technique
* Boundary Value Technique
* State Transition Technique
* Equivalence partitioning
* Graph-Based Testing
* Error Guessing Technique.

**Decision Table Technique:**

Decision Table Technique is a systematic approach where various input combinations and their respective system behaviour are captured in a tabular form. It is appropriate for the functions that have a logical relationship between two and more than two inputs.

**Boundary Value Technique:**

Boundary Value Technique is used to test boundary values, boundary values are those that contain the upper and lower limit of a variable. It tests, while entering boundary value whether the software is producing correct output or not.

**State Transition Technique:**

State Transition Technique is used to capture the behaviour of the software application when different input values are given to the same function. This applies to those types of applications that provide the specific number of attempts to access the application.

**Equivalence partitioning**

Equivalence partitioning is a technique of software testing in which input data divided into partitions of valid and invalid values, and it is mandatory that all partitions must exhibit the same behaviour.

**Graph-Based Testing:**

It is similar to a decision-based test case design approach where the relationship between

links and input cases are considered**.**

**Error Guessing Technique.**

Error guessing is a technique in which there is no specific method for identifying the error. It is based on the experience of the test analyst, where the tester uses the experience to guess the problematic areas of the software.

**Testing**

We follow manual testing here for the application, a type of software testing where testers

manually write and execute test cases without using any automation tools. This method is the most primitive type and helps in finding bugs in the application code, ensuring that application is error free and is deliverable to the user/customer. Different Levels of Manual Testing Types of manual testing are:

* Unit Testing
* Integration Testing
* System Testing
* Acceptance Testing
* Black-box Testing
* White-box Testing

**Black-Box Testing**

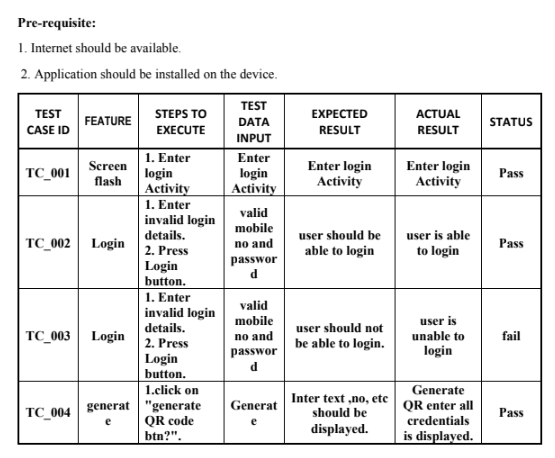
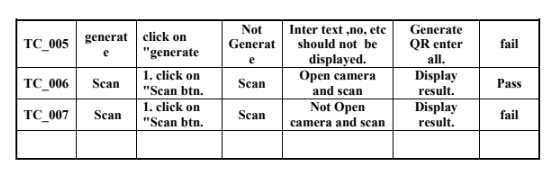
As mentioned above, Black box testing is the testing method which is used to test the software without any knowledge of code or program. In short, the code under test environment is called as “Black-Box”. When performing the testing, the tester expects particular output/results, but not aware of how the application actually processes internally. Since this method is considered as the last step, it is to check whether the application is working as per user expectations.

**Advantages of Black Box Testing:**

* Efficient when used on large systems.
* Since the tester and developer are independent of each other, testing is balanced and unprejudiced.
* Tester can be non-technical.
* There is no need for the tester to have detailed functional knowledge of system.
* Testing helps to identify vagueness and contradictions in functional specifications.
* Test cases can be designed as soon as the functional specifications are complete.

**Disadvantages of Black Box Testing:**

* Test cases are challenging to design without having clear functional specifications.
* It is difficult to identify tricky inputs if the test cases are not developed based on specifications.
* It is difficult to identify all possible inputs in limited testing time. As a result, writing test cases may be slow and difficult.
* There are chances of having unidentified paths during the testing process.
* There is a high probability of repeating tests already performed by the programmer.



**⚫ Conclusion: -**

**Black box testing helps** to find the gaps **in functionality, usability, and other features**. This form of testing gives an overview of software performance and its output. It improves software quality and reduces the time to market. This form of testing mitigates the risk of software failures at the user’s end.

**Experiment No: 08**

**⚫Aim: Write test cases for White box testing.**

**⚫Theory:**

**⚫ Project Name: - The QR CODE SCANNER**

**White Box Testing:**

White Box Testing is a testing technique in which software’s internal structure, design, and coding are tested to verify input-output flow and improve design, usability, and security. In white box testing, code is visible to testers, so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing, and Glass box testing.

**The white box testing contains various tests, which are as follows:**

* Path testing
* Loop testing
* Condition testing
* Testing based on the memory perspective
* Test performance of the program.

**Techniques Used in White Box Testing:**

* Data Flow Testing
* Control Flow Testing
* Branch Testing
* Statement Testing
* Decision Testing

**Data Flow Testing:**

Data flow testing is a group of testing strategies that examines the control flow of programs in order to explore the sequence of variables according to the sequence of events.

**Control Flow Testing:**

Control flow testing determines the execution order of statements or instructions of the program through a control structure. The control structure of a program is used to develop a test case for the program. In this technique, a particular part of a large program is selected by the tester to set the testing path. Test cases represented by the control graph of the program.

**Branch coverage technique:**

Branch coverage technique is used to cover all branches of the control flow graph. It covers all the possible outcomes (true and false) of each condition of decision point at least once.

**Statement coverage technique:**

Statement coverage technique is used to design white box test cases. This technique involves execution of all statements of the source code at least once. It is used to calculate the total number of executed statements in the source code, out of total statements present in the source code.

**Decision Testing:**

This technique reports true and false outcomes of Boolean expressions. Whenever there is a possibility of two or more outcomes from the statements like do while statement, if statement and case statement (Control flow statements), it is considered as decision point because there are two outcomes either true or false.

Test Case 1:

This case the logo screen flash done move next page

  new Handler().postDelayed(new Runnable() {

// Using handler with postDelayed called runnable run method

  @Override

     public void run() {

     Intent i = new Intent(MainSplashScreen.this, FirstScreen.class);

      startActivity(i);

 // close this activity

      finish();

         }

       }, 5\*1000); // wait for 5 seconds

Test Case 2: Login authentication+add databsae

1. private void initView() {
2. mCallbacks = new PhoneAuthProvider.OnVerificationStateChangedCallbacks() {
3. @Override
4. public void onCodeAutoRetrievalTimeOut(String verificationId) {
   1. if (progressDialog != null) {
   2. dismissProgressDialog(progressDialog);
   3. }
   4. notifyUserAndRetry("Your Phone Number Verification is failed.Retry again!");
5. }
6. @Override
7. public void onVerificationCompleted(PhoneAuthCredential credential) {
   1. Log.d("onVerificationCompleted", "onVerificationCompleted:" + credential);
   2. if (progressDialog != null) {
   3. dismissProgressDialog(progressDialog);
   4. }
   5. signInWithPhoneAuthCredential(credential);
8. }
9. @Override
10. public void onVerificationFailed(FirebaseException e) {
    1. Log.w("onVerificationFailed", "onVerificationFailed", e);
    2. if (progressDialog != null) {
    3. dismissProgressDialog(progressDialog);
    4. }
    5. if (e instanceof FirebaseAuthInvalidCredentialsException) {
    6. Log.e("Exception:", "FirebaseAuthInvalidCredentialsException" + e);
    7. } else if (e instanceof FirebaseTooManyRequestsException) {
    8. Log.e("Exception:", "FirebaseTooManyRequestsException" + e);
    9. }
    10. notifyUserAndRetry("Your Phone Number Verification is failed.Retry again!");
11. }
12. @Override
13. public void onCodeSent(String verificationId,
    * 1. PhoneAuthProvider.ForceResendingToken token) {
    1. Log.d("onCodeSent", "onCodeSent:" + verificationId);
    2. Log.i("Verification code:", verificationId);
14. }
15. };

Test case 03 : scan QR code:

1. **public** **class** ScannedBarcodeActivity **extends** AppCompatActivity {

4. SurfaceView surfaceView;
5. TextView txtBarcodeValue;
6. **private** BarcodeDetector barcodeDetector;
7. **private** CameraSource cameraSource;
8. **private** **static** **final** **int** REQUEST\_CAMERA\_PERMISSION = 201;
9. Button btnAction;
10. String intentData = "";
11. **boolean** isEmail = **false**;
13. @Override
14. **protected** **void** onCreate(Bundle savedInstanceState) {
15. **super**.onCreate(savedInstanceState);
16. setContentView(R.layout.activity\_scanned\_barcode);
17. initViews();
18. }
20. **private** **void** initViews() {
21. txtBarcodeValue = findViewById(R.id.txtBarcodeValue);
22. surfaceView = findViewById(R.id.surfaceView);
23. btnAction = findViewById(R.id.btnAction);
24. btnAction.setOnClickListener(**new** View.OnClickListener() {
25. @Override
26. **public** **void** onClick(View v) {
27. **if** (intentData.length() > 0) {
28. **if** (isEmail)
29. startActivity(**new** Intent(ScannedBarcodeActivity.**this**, EmailActivity.**class**).putExtra("email\_address", intentData));
30. **else** {
31. startActivity(**new** Intent(Intent.ACTION\_VIEW, Uri.parse(intentData)));
32. }
33. }
34. }
35. });
36. }
38. **private** **void** initialiseDetectorsAndSources() {
40. Toast.makeText(getApplicationContext(), "Barcode scanner started", Toast.LENGTH\_SHORT).show();
41. barcodeDetector = **new** BarcodeDetector.Builder(**this**)
42. .setBarcodeFormats(Barcode.ALL\_FORMATS)
43. .build();
45. cameraSource = **new** CameraSource.Builder(**this**, barcodeDetector)
46. .setRequestedPreviewSize(1920, 1080)
47. .setAutoFocusEnabled(**true**) //you should add this feature
48. .build();
50. surfaceView.getHolder().addCallback(**new** SurfaceHolder.Callback() {
51. @Override
52. **public** **void** surfaceCreated(SurfaceHolder holder) {
53. **try** {
54. **if** (ActivityCompat.checkSelfPermission(ScannedBarcodeActivity.**this**, Manifest.permission.CAMERA) == PackageManager.PERMISSION\_GRANTED) {
55. cameraSource.start(surfaceView.getHolder());
56. } **else** {
57. ActivityCompat.requestPermissions(ScannedBarcodeActivity.**this**, **new**
58. String[]{Manifest.permission.CAMERA}, REQUEST\_CAMERA\_PERMISSION);
59. }
61. } **catch** (IOException e) {
62. e.printStackTrace();
63. }
64. }
66. @Override
67. **public** **void** surfaceChanged(SurfaceHolder holder, **int** format, **int** width, **int** height) {
68. }
70. @Override
71. **public** **void** surfaceDestroyed(SurfaceHolder holder) {
72. cameraSource.stop();
73. }
74. });

77. barcodeDetector.setProcessor(**new** Detector.Processor<Barcode>() {
78. @Override
79. **public** **void** release() {
80. Toast.makeText(getApplicationContext(), "To prevent memory leaks barcode scanner has been stopped", Toast.LENGTH\_SHORT).show();
81. }
83. @Override
84. **public** **void** receiveDetections(Detector.Detections<Barcode> detections) {
85. **final** SparseArray<Barcode> barcodes = detections.getDetectedItems();
86. **if** (barcodes.size() != 0) {
87. txtBarcodeValue.post(**new** Runnable() {
88. @Override
89. **public** **void** run() {
91. **if** (barcodes.valueAt(0).email != **null**) {
92. txtBarcodeValue.removeCallbacks(**null**);
93. intentData = barcodes.valueAt(0).email.address;
94. txtBarcodeValue.setText(intentData);
95. isEmail = **true**;
96. btnAction.setText("ADD CONTENT TO THE MAIL");
97. } **else** {
98. isEmail = **false**;
99. btnAction.setText("LAUNCH URL");
100. intentData = barcodes.valueAt(0).displayValue;
101. txtBarcodeValue.setText(intentData);
102. }
103. }
104. });
105. }
106. }
107. });
108. }

111. @Override
112. **protected** **void** onPause() {
113. **super**.onPause();
114. cameraSource.release();
115. }
117. @Override
118. **protected** **void** onResume() {
119. **super**.onResume();
120. initialiseDetectorsAndSources();
121. }
122. }

**Advantages of White box testing:**

* White box testing optimizes code so hidden errors can be identified.
* Test cases of white box testing can be easily automated.
* This testing is more thorough than other testing approaches as it covers all code paths.
* It can be started in the SDLC phase even without GUI.

**Disadvantages of White box testing:**

* White box testing is too much time consuming when it comes to large-scale programming applications.
* White box testing is much expensive and complex.
* It can lead to production error because it is not detailed by the developers.
* White box testing needs professional programmers who have a detailed knowledge and understanding of programming language and implementation.

**⚫ Conclusion: -**

White box testing for security is useful and effective. It should follow a risk-based approach to balance the testing effort with consequences of software failure. Architectural and design-level risk analysis provide the right context to plan and perform white box testing. White box testing can be used with black box testing to improve overall test effectiveness. It uncovers programming and implementation errors.

**Experiment No: 09**

**⚫Aim: Preparation of Risk Mitigation, Monitoring, and Management Plan (RMMM).**

**⚫Theory:**

**⚫ Project Name: - The QR CODE SCANNER**

**Risk Management**

A software project can be concerned with a large variety of risks. In order to be adept to systematically identify the significant risks which might affect a software project, it is essential to classify risks into different classes. The project manager can then check which risks from each class are relevant to the project.

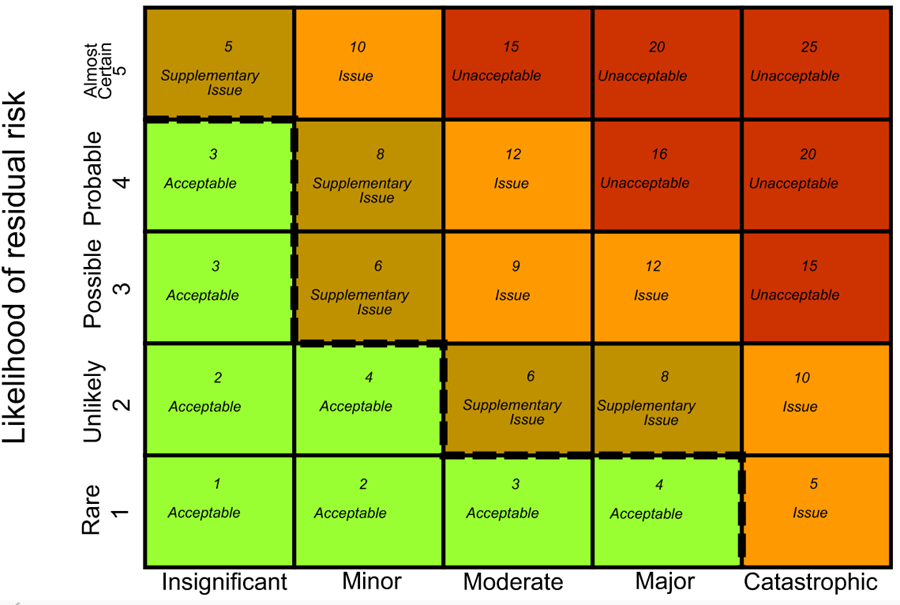
**There are three main classifications of risks which can affect a software project:**

* Project risks
* Technical risks
* Business risks

1. **Project risks**: Project risks concern differ forms of budgetary, schedule, personnel, resource, and customer-related problems. A vital project risk is schedule slippage. Since the software is intangible, it is very tough to monitor and control a software project. It is very tough to control something which cannot be identified. For any manufacturing program, such as the manufacturing of cars, the plan executive can recognize the product taking shape.

2. **Technical risks**: Technical risks concern potential method, implementation, interfacing, testing, and maintenance issue. It also consists of an ambiguous specification, incomplete specification, changing specification, technical uncertainty, and technical obsolescence. Most technical risks appear due to the development team's insufficient knowledge about the project.

3. **Business risks**: This type of risks contain risks of building an excellent product that no one need, losing budgetary or personnel commitments, etc.



**RMMM Plan:**

A risk management technique is usually seen in the software Project plan. This can be divided into Risk Mitigation, Monitoring, and Management Plan (RMMM). In this plan, all works are done as part of risk analysis. As part of the overall project plan project manager generally uses this RMMM plan.

In some software teams, risk is documented with the help of a Risk Information Sheet (RIS). This RIS is controlled by using a database system for easier management of information i.e creation, priority ordering, searching, and other analysis. After documentation of RMMM and start of a project, risk mitigation and monitoring steps will start.

**Risk Mitigation:**

It is an activity used to avoid problems (Risk Avoidance).

Steps for mitigating the risks as follows.

* Finding out the risk.
* Removing causes that are the reason for risk creation.
* Controlling the corresponding documents from time to time.
* Conducting timely reviews to speed up the work.

**Risk Monitoring:**

It is an activity used for project tracking.

It has the following primary objectives as follows

* To check if predicted risks occur or not.
* To ensure proper application of risk aversion steps defined for risk.
* To collect data for future risk analysis.
* To allocate what problems are caused by which risks throughout the project.

**Risk Management and planning:**

It assumes that the mitigation activity failed and the risk is a reality. This task is done by Project manager when risk becomes reality and causes severe problems. If the project manager effectively uses project mitigation to remove risks successfully then it is easier to manage the risks. This shows that the response that will be taken for each risk by a manager. The main objective of the risk management plan is the risk register. This risk register describes and focuses on the predicted threats to a software project.

**Risk Mitigation:**

To mitigate this risk, project management must develop a strategy for reducing turnover. The possible steps to be taken are:

* Meet the current staff to determine causes for turnover (e.g., poor working conditions, low pay, competitive job market).
* Mitigate those causes that are under our control before the project starts.
* Once the project commences, assume turnover will occur and develop techniques to ensure continuity when people leave.
* Organize project teams so that information about each development activity is widely dispersed.
* Define documentation standards and establish mechanisms to ensure that documents are developed in a timely manner.
* Assign a backup staff member for every critical technologist.

**Risk Monitoring:**

As the project proceeds, risk monitoring activities commence. The project manager monitors factors that may provide an indication of whether the risk is becoming more or less likely. In the case of high staff turnover, the following factors can be monitored:

* General attitude of team members based on project pressures.
* Interpersonal relationships among team members.
* Potential problems with compensation and benefits.
* The availability of jobs within the company and outside it.

**Risk Management:**

Risk management and contingency planning assumes that mitigation efforts have failed and that the risk has become a reality. Continuing the example, the project is well underway, and a number of people announce that they will be leaving. If the mitigation strategy has been followed, backup is available, information is documented, and knowledge has been dispersed across the team. In addition, the project manager may temporarily refocus resources (and readjust the project schedule) to those functions that are fully staffed, enabling newcomers who must be added to the team to “get up to the speed “.

**Risk Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SR NO.** | **Risk Type** | **Risk identification** | **Migration strategies** | **Severity** |
| 01 | Constriction | Planning | We build the app within planning | marginal |
| 02 | budget | Cost | We create app over budget | marginal |
| 03 | Market | Market | We introduce the market level but some level of computation is already available. | negligible |
| 04 |  | Performance | The OR app is performance is faster than another app | critical |
| 05 |  | Operational | The OR app is function is faster than another app | catastrophic |
| 06 | Plan | strategic | We create strategic risk to build app | marginal |
|  |  |  |  |  |
| 07 | Time | Schedule Risk | The app is build within the given Schedule | critical |
| 08 | Deployment | Logical Risk | The app gives as we test and after deployment the client | catastrophic |
| 09 | Feedback | Maintained Risk | After The give the valid reason for app we finding in risk | catastrophic |

**⚫ Conclusion: -** We learned about **risk management activities** and the **RMMM** plan in this article. We also infer from this article how project managers do risk management, and the RMMM plan is one of these techniques.

**Experiment No: 10**

**⚫Aim: Version Controlling of the project.**

**⚫Theory:**

**⚫ Project Name: - The QR CODE SCANNER**

**Version control system:**

A version control system is a software that tracks changes to a file or set of files over time so that you can recall specific versions later. It also allows you to work together with other programmers. The version control system is a collection of software tools that help a team to manage changes in a source code. It uses a special kind of database to keep track of every modification to the code. Developers can compare earlier versions of the code with an older version to fix the mistakes.

**Benefits of the Version Control System**

The Version Control System is very helpful and beneficial in software development; developing software without using version control is unsafe. It provides backups for uncertainty. Version control systems offer a speedy interface to developers. It also allows software teams to preserve efficiency and agility according to the team scales to include more developers.

Some key benefits of having a version control system are as follows.

* Complete change history of the file
* Simultaneously working
* Branching and merging
* Traceability

**repository** – single location where the current and all prior versions of the files are stored

**working copy** – the local copy of a file from the repository which can be modified and then checked in or “committed” to the repository

**check-out** – the process of creating a working copy from the repository (either the current version or an earlier version)

**check-in** – a check-in or commit occurs when changes made to a working copy are merged into the repository

**diff** – a summary of the differences between a working copy and a file in the repository, often taking the form of the two files side-by-side with differences highlighted

**conflict** – a conflict occurs when two or more developers attempt to make changes to the

same file and the system is unable to reconcile the changes (note: conflicts generally must be resolved by either choosing one version over the other or by integrating the changes from both into the repository by hand)

**update** – merges recent changes to the repository into a working copy

**Types of Version Control System:**

* Localized version Control System
* Centralized version control systems
* Distributed version control systems

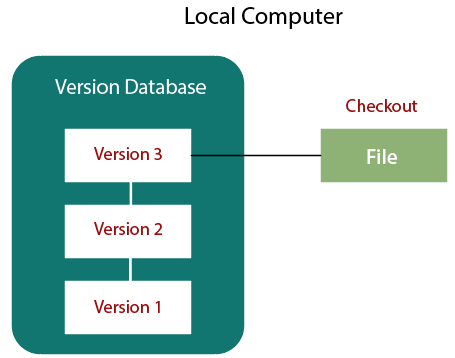
**localized version control:**

The localized version control method is a common approach because of its simplicity.

But this approach leads to a higher chance of error. In this approach, you may forget

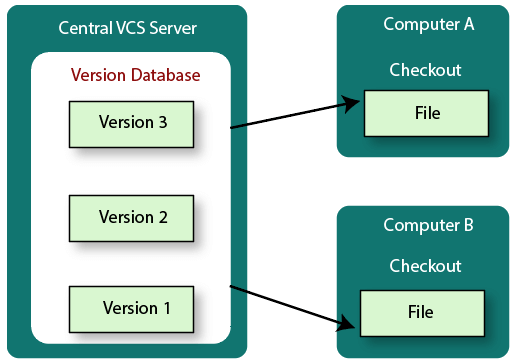
which directory you’re in and accidentally write to the wrong file or copy over files you

don want to.



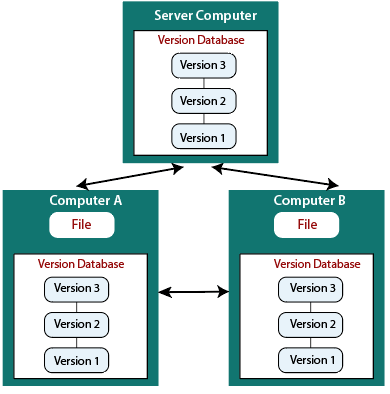
**Central version control:**

The developers needed to collaborate with other developers on other systems. The localized version control system failed in this case. To deal with this problem, Centralized Version Control Systems were developed.



**Distributed version control:**

In a Distributed Version Control System (such as Git, Mercurial, Bazaar or Darcs), the user has a local copy of a repository. So, the clients it just checks out the latest snapshot of the files even they can fully mirror the repository. The local repository contains all the files and metadata present in the main repository.

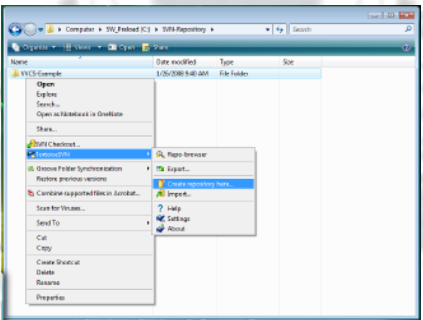


The basic steps that one would use to get started with a **version control tool** are as follows:

1. Create a repository
2. Import a directory structure and/or files into the repository
3. Check-out the repository version as a working copy
4. Edit/modify the files in the working copy and examine the differences
5. between the working copy and the repository (i.e., diff)
6. Check-in (or commit) the changes to the repository.

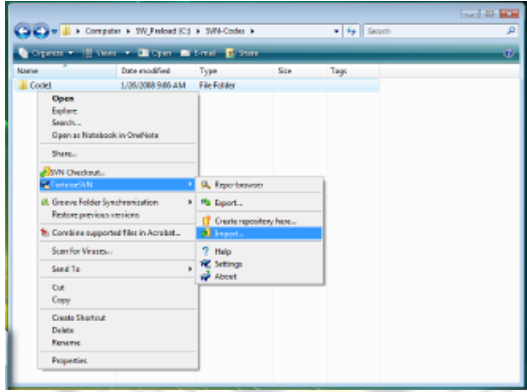
1**. Creating a Repository:**

Determine a location for the repository, ideally on a server which is automatically backed up. Create a folder with the name of the repository; in this example the repository is called “VVCSExample.” Right click on the folder name, choose “TortoiseSVN” (which is integrated into the Microsoft Windows Explorer menu), then “Create Repository Here.” Choose the Native Filesystem, then you should see the message “Repository Successfully Created.



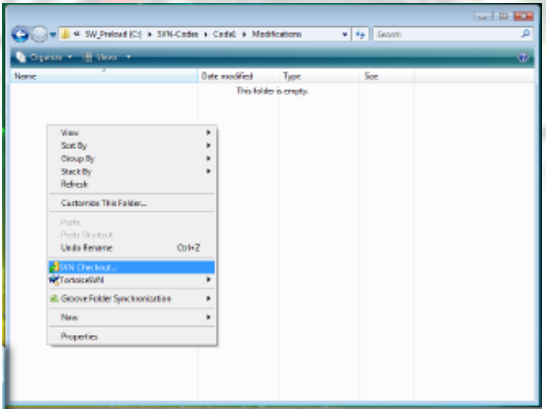
**2. Importing a File into the Repository**

Right click on the directory containing the file(s) and/or directory structure you wish to import to the repository (note, the directory that you click on will not be imported). Here we will simply be importing the file “code1.f” from directory “Code1.” This code creates a 17×17 two-dimensional Cartesian grid for x and y between 0 and 1. Browse until you find the location of the repository “VVCS-Example” and select that directory name. This version of the code will be Revision 1.



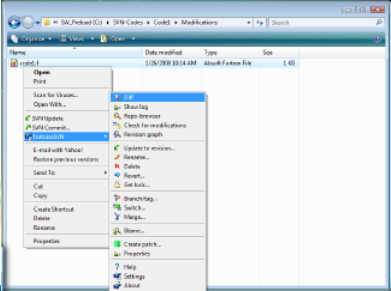
**3. Checking the Code out from the Repository**

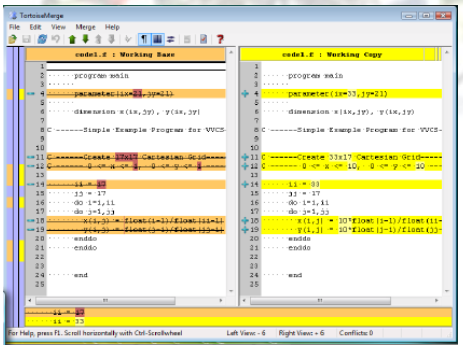
You now have the code “code1.f” safely placed in the repository. To modify this code and create a new revision, you will need to check out a working copy of the code. Go to the directory where you will be modifying the code, in this example, the directory “Modifications.” Right click in Windows Explorer, and select “SVN Checkout…” Select the name of the repository you just created, then click “OK.” You will now get a window telling you that you are at Revision 1. Notice the green check mark on the “code1.f” icon. This indicates that this working copy is up to date with the version in the repository.



**4. Modify the Code and Compare to the Repository Version**

The code “code1.f” can now be modified. Here we will change the code to allow the Cartesian grid to contain 33x17 points between the values of zero and ten. Once the code has been modified, you will notice that the green check mark has been replaced by a red exclamation point, indicating that the current working copy has been modified from the version in the repository. To examine these differences, right click on the “code1.f” file, select “TortoiseSVN, “then “Diff.” This opens the “Tortoise Merge” tool which clearly shows the modifications to the repository version (Working Base) that were made in the Working Copy.





**⚫ Conclusion:** We, understood the basics of **Version Control** and the **benefits** of **Version Control**. Now, you know why this system is so widely used and why small or big company opts for Version Control.

**Viva Questions**

**Introduction To Software Engineering and Process Models**

Software Engineering is an engineering branch related to the evolution of software product using well-defined scientific principles, techniques, and procedures. The result of software engineering is an effective and reliable software product.

Need of Software Engineering

* Huge Programming
* Quality Management
* Adaptability
* Cost
* Dynamic Nature

**software life cycle model**

A software life cycle model (also termed process model) is a pictorial and diagrammatic representation of the software life cycle. A life cycle model represents all the methods required to make a software product transit through its life cycle stages. It also captures the structure in which these methods are to be undertaken.



**Software Process Framework** is an abstraction of the software development process. It details the steps and chronological order of a process. Since it serves as a foundation for them, it is utilized in most applications. Task sets, umbrella activities, and process framework activities all define the characteristics of the software development process.

Software process includes:

Tasks – focus on a small, specific objective.

Action – set of tasks that produce a major work product.

Activities – group of related tasks and actions for a major objective.

**What is Capability Maturity Model (CMM) Levels?**

The Software Engineering Institute (SEI) Capability Maturity Model (CMM) specifies an increasing series of levels of a software development organization. The higher the level, the better the software development process, hence reaching each level is an expensive and time-consuming process.

1. Initial
2. Repeatable/Managed
3. Defined
4. Quantitatively Managed
5. Optimizing

**Prescriptive Process Models: The Waterfall, Incremental Process Models,**

**Evolutionary Process Models: RAD & Spiral**

**Waterfall Model**

* The Waterfall Model was the first Process Model to be introduced.
* It is also referred to as a linear-sequential life cycle model.
* It is very simple to understand and use.
* In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.



**Some of the major advantages of the Waterfall Model are as follows −**

* Simple and easy to understand and use
* Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
* Phases are processed and completed one at a time.
* Works well for smaller projects where requirements are very well understood.
* Clearly defined stages.
* Well understood milestones.
* Easy to arrange tasks.

**The major disadvantages of the Waterfall Model are as follows −**

* No working software is produced until late during the life cycle.
* High amounts of risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
* It is difficult to measure progress within stages.
* Cannot accommodate changing requirements.

**RAD**

RAD is a linear sequential software development process model that emphasizes a concise development cycle using an element-based construction approach. If the requirements are well understood and described, and the project scope is a constraint, the RAD process enables a development team to create a fully functional system within a concise time period.

**WE USE IN:**

* When the system should need to create the project that modularizes in a short span time (2-3 months).
* When the requirements are well-known.
* When the technical risk is limited.
* When there's a necessity to make a system, which modularized in 2-3 months of period.
* It should be used only if the budget allows the use of automatic code generating tools.

**Advantage of RAD Model**

* This model is flexible for change.
* In this model, changes are adoptable.
* Each phase in RAD brings highest priority functionality to the customer.
* It reduced development time.

**Disadvantage of RAD Model**

* It required highly skilled designers.
* All application is not compatible with RAD.
* For smaller projects, we cannot use the RAD model.
* On the high technical risk, it's not suitable

**Incremental Model:**

* **Incremental Model** is a process of software development where requirements divided into multiple standalone modules of the software development cycle.
* In this model, each module goes through the requirements, design, implementation and testing phases.
* Every subsequent release of the module adds function to the previous release. The process continues until the complete system achieved.
* When the requirements are superior.
* A project has a lengthy development schedule.
* When Software team are not very well skilled or trained.
* When the customer demands a quick release of the product.
* You can develop prioritized requirements first.

**Advantage of Incremental Model**

* Errors are easy to be recognized.
* Easier to test and debug
* More flexible.
* Simple to manage risk because it handled during its iteration.
* The Client gets important functionality early.

**Disadvantage of Incremental Model**

* Need for good planning
* Total Cost is high.
* Well defined module interfaces are needed.

**The spiral model**

* **The spiral model**, initially proposed by Boehm, is an evolutionary software process model that couples the iterative feature of prototyping with the controlled and systematic aspects of the linear sequential model.
* It implements the potential for rapid development of new versions of the software.
* Using the spiral model, the software is developed in a series of incremental releases. During the early iterations, the additional release may be a paper model or prototype.
* During later iterations, more and more complete versions of the engineered system are produced.

## **When to use Spiral Model?**

* When deliverance is required to be frequent.
* When the project is large
* When requirements are unclear and complex
* When changes may require at any time
* Large and high budget projects

## **Advantages**

* High amount of risk analysis
* Useful for large and mission-critical projects.

## **Disadvantages**

* Can be a costly model to use.
* Risk analysis needed highly particular expertise
* Doesn't work well for smaller projects.

**Agile methods**

Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning

The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance. WHEN USE:

* When frequent changes are required.
* When a highly qualified and experienced team is available.
* When a customer is ready to have a meeting with a software team all the time.
* When project size is small.

**Phases of Agile Model:**

Following are the phases in the Agile model are as follows:

* Requirements gathering
* Design the requirements
* Construction/ iteration
* Testing/ Quality assurance
* Deployment
* Feedback

Agile Testing Methods:

* Scrum
* Crystal
* Dynamic Software Development Method(DSDM)
* Feature Driven Development(FDD)
* Lean Software Development
* eXtreme Programming(XP)

**Advantage(Pros) of Agile Method:**

* Frequent Delivery
* Face-to-Face Communication with clients.
* Efficient design and fulfils the business requirement.
* Anytime changes are acceptable.
* It reduces total development time.

**The disadvantages of the Agile Model are as follows −**

* Not suitable for handling complex dependencies.
* More risk of sustainability, maintainability and extensibility.
* An overall plan, an agile leader and agile PM practice is a must without which it will not work.
* Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.

**Software Requirements Analysis and Modeling**

Analysis Model is a technical representation of the system. It acts as a link between system description and design model.

In Analysis Modelling, information, behaviour, and functions of the system are defined and translated into the architecture, component, and interface level design in the design modeling**.**

**Data Dictionary:**

It is a repository that consists of a description of all data objects used or produced by the software. It stores the collection of data present in the software. It is a very crucial element of the analysis model.

**Entity Relationship Diagram (ERD):**

It depicts the relationship between data objects and is used in conducting data modeling activities. The attributes of each object in the Entity-Relationship Diagram can be described using Data object description. It provides the basis for activity related to data design.

**Data Flow Diagrams**

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

Level 0,1,2

**Scenario-based elements :**

Using a scenario-based approach, system is described from user’s point of view. For example, basic use cases and their corresponding use-case diagrams evolve into more elaborate template-based use cases. Figure 1(a) depicts a UML activity diagram for eliciting requirements and representing them using use cases. There are three levels of elaboration.

**Software Requirement Specification (SRS):**

**Software Requirement Specification (SRS)** Format as name suggests, is complete specification and description of requirements of software that needs to be fulfilled for successful development of software system. These requirements can be functional as well as non-functional depending upon type of requirement. The interaction between different customers and contractor is done because its necessary to fully understand needs of customers.

1. Introduction

* Purpose of this document
* Scope of this document
* Overview

2. General description

3. Functional Requirements

4. Interface Requirements

5. Performance Requirements

6. Design Constraints

7. Non-Functional Attributes

8. Preliminary Schedule and Budget

9. Appendices

**Software Estimation Metrics**

**A software metric**

A software metric is a measure of software characteristics which are measurable or countable. Software metrics are valuable for many reasons, including measuring software performance, planning work items, measuring productivity, and many other uses.

**Classification of Software Metrics:**

There are 3 types of software metrics:

Product Metrics: Product metrics are used to evaluate the state of the product, tracing risks and undercover prospective problem areas. The ability of the team to control quality is evaluated.

Process Metrics: Process metrics pay particular attention to enhancing the long-term process of the team or organization.

Project Metrics: The project matrix describes the project characteristic and execution process.

* Number of software developer
* Staffing patterns over the life cycle of software
* Cost and schedule
* Productivity

**Advantages of Software Metrics :**

* Reduction in cost or budget.
* It helps to identify the particular area for improvising.
* It helps to increase the product quality.
* Managing the workloads and teams.
* Reduction in overall time to produce the product,.
* It helps to determine the complexity of the code and to test the code with resources.
* It helps in providing effective planning, controlling and managing of the entire product.

**Disadvantages of Software Metrics :**

* It is expensive and difficult to implement the metrics in some cases.
* Performance of the entire team or an individual from the team can’t be determined. Only the performance of the product is determined.
* Sometimes the quality of the product is not met with the expectation.
* It leads to measure the unwanted data which is wastage of time.
* Measuring the incorrect data leads to make wrong decision making.

**Estimation project management:**

Estimation of the size of the software is an essential part of Software Project Management. It helps the project manager to further predict the effort and time which will be needed to build the project.

**Lines of Code (LOC):** As the name suggests, LOC count the total number of lines of source code in a project. The units of LOC are:

* KLOC- Thousand lines of code
* NLOC- Non-comment lines of code
* KDSI- Thousands of delivered source instruction.

**Advantages**:

* Universally accepted and is used in many models like COCOMO.
* Estimation is closer to the developer’s perspective.
* Simple to use.

**Disadvantages**:

* Different programming languages contain a different number of lines.
* No proper industry standard exists for this technique.
* It is difficult to estimate the size using this technique in the early stages of the project.

Cocomo model

**Project Scheduling**

Project-task scheduling is a significant project planning activity. It comprises deciding which functions would be taken up when. To schedule the project plan, a software project manager wants to do the following:

* Identify all the functions required to complete the project.
* Break down large functions into small activities.
* Determine the dependency among various activities.
* Establish the most likely size for the time duration required to complete the activities.
* Allocate resources to activities.
* Plan the beginning and ending dates for different activities.
* Determine the critical path. A critical way is the group of activities that decide the duration of the project.

**Advantages of Project Scheduling:**

There are several advantages provided by project schedule in our project management:

* It simply ensures that everyone remains on same page as far as tasks get completed, dependencies, and deadlines.
* It helps in identifying issues early and concerns such as lack or unavailability of resources.
* It also helps to identify relationships and to monitor process.
* It provides effective budget management and risk mitigation.

**TRACK:**

* Gannt chart
* PERT chart

| S. No. | Black Box Testing | White Box Testing |
| --- | --- | --- |
| 1. | It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it. | It is a way of testing the software in which the tester has knowledge about the internal structure or the code or the program of the software. |
| 2. | Implementation of code is not needed for black box testing. | Code implementation is necessary for white box testing. |
| 3. | It is mostly done by software testers. | It is mostly done by software developers. |
| 4. | No knowledge of implementation is needed. | Knowledge of implementation is required. |
| 5. | It can be referred to as outer or external software testing. | It is the inner or the internal software testing. |
| 6. | It is a functional test of the software. | It is a structural test of the software. |
| 7. | This testing can be initiated based on the requirement specifications document. | This type of testing of software is started after a detail design document. |
| 8. | No knowledge of programming is required. | It is mandatory to have knowledge of programming. |
| 9. | It is the behaviour testing of the software. | It is the logic testing of the software. |
| 10. | It is applicable to the higher levels of testing of software. | It is generally applicable to the lower levels of software testing. |
| 11. | It is also called closed testing. | It is also called as clear box testing. |
| 12. | It is least time consuming. | It is most time consuming. |
| 13. | It is not suitable or preferred for algorithm testing. | It is suitable for algorithm testing. |
| 14. | Can be done by trial and error ways and methods. | Data domains along with inner or internal boundaries can be better tested. |
| 15. | **Example:** Search something on google by using keywords | **Example:** By input to check and verify loops |
| 16. | **Black-box test design techniques-**   * Decision table testing * All-pairs testing * Equivalence partitioning * Error guessing | **White-box test design techniques-**   * Control flow testing * Data flow testing * Branch testing |
| 17. | **Types of Black Box Testing:**   * Functional Testing * Non-functional testing * Regression Testing | **Types of White Box Testing:**   * Path Testing * Loop Testing * Condition testing |
| 18. | It is less exhaustive as compared to white box testing. | It is comparatively more exhaustive than black box testing. |

**What is a “version control system”?**

Version control systems are a category of software tools that helps in recording changes made to files by keeping a track of modifications done in the code.

**Benefits of the version control system:**

* Enhances the project development speed by providing efficient collaboration,
* Leverages the productivity, expedites product delivery, and skills of the employees through better communication and assistance,
* Reduce possibilities of errors and conflicts meanwhile project development through traceability to every small change,
* Employees or contributors of the project can contribute from anywhere irrespective of the different geographical locations through this VCS,
* For each different contributor to the project, a different working copy is maintained and not merged to the main file unless the working copy is validated. The most popular example is Git, Helix core, Microsoft TFS,

Helps in recovery in case of any disaster or contingent situation,

* Informs us about Who, What, When, Why changes have been made.

**Types of Version Control Systems:**

* Local Version Control Systems
* Centralized Version Control Systems
* Distributed Version Control Systems

**Purpose of Version Control:**

* Multiple people can work simultaneously on a single project. Everyone works on and edits their own copy of the files and it is up to them when they wish to share the changes made by them with the rest of the team.
* It also enables one person to use multiple computers to work on a project, so it is valuable even if you are working by yourself.
* It integrates the work that is done simultaneously by different members of the team. In some rare cases, when conflicting edits are made by two people to the same line of a file, then human assistance is requested by the version control system in deciding what should be done.

**RMMM Plan :**

A risk management technique is usually seen in the software Project plan. This can be divided into Risk Mitigation, Monitoring, and Management Plan (RMMM). In this plan, all works are done as part of risk analysis. As part of the overall project plan project manager generally uses this RMMM plan.

**Risk Mitigation :**

It is an activity used to avoid problems (Risk Avoidance).

Steps for mitigating the risks as follows.

* Finding out the risk.
* Removing causes that are the reason for risk creation.
* Controlling the corresponding documents from time to time.
* Conducting timely reviews to speed up the work.

**Risk Monitoring:**

It is an activity used for project tracking.

It has the following primary objectives as follows.

To check if predicted risks occur or not.

* To ensure proper application of risk aversion steps defined for risk.
* To collect data for future risk analysis.
* To allocate what problems are caused by which risks throughout the project.

**Risk Management and planning :**

It assumes that the mitigation activity failed and the risk is a reality. This task is done by Project manager when risk becomes reality and causes severe problems. If the project manager effectively uses project mitigation to remove risks successfully then it is easier to manage the risks. This shows that the response that will be taken for each risk by a manager. The main objective of the risk management plan is the risk register. This risk register describes and focuses on the predicted threats to a software project.

Example:

Let us understand RMMM with the help of an example of high staff turnover.

**Risk Mitigation:**

* To mitigate this risk, project management must develop a strategy for reducing turnover. The possible steps to be taken are:
* Meet the current staff to determine causes for turnover (e.g., poor working conditions, low pay, competitive job market).
* Mitigate those causes that are under our control before the project starts.
* Once the project commences, assume turnover will occur and develop techniques to ensure continuity when people leave.
* Organize project teams so that information about each development activity is widely dispersed.
* Define documentation standards and establish mechanisms to ensure that documents are developed in a timely manner.
* Assign a backup staff member for every critical technologist.

**Risk Monitoring:**

As the project proceeds, risk monitoring activities commence. The project manager monitors factors that may provide an indication of whether the risk is becoming more or less likely. In the case of high staff turnover, the following factors can be monitored:

* General attitude of team members based on project pressures.
* Interpersonal relationships among team members.
* Potential problems with compensation and benefits.
* The availability of jobs within the company and outside it.

**Risk Management:**

Risk management and contingency planning assumes that mitigation efforts have failed and that the risk has become a reality. Continuing the example, the project is well underway, and a number of people announce that they will be leaving. If the mitigation strategy has been followed, backup is available, information is documented, and knowledge has been dispersed across the team. In addition, the project manager may temporarily refocus resources (and readjust the project schedule) to those functions that are fully staffed, enabling newcomers who must be added to the team to “get up to the speed“.

Drawbacks of RMMM:

* It incurs additional project costs.
* It takes additional time.
* For larger projects, implementing an RMMM may itself turn out to be another tedious project.
* RMMM does not guarantee a risk-free project, in fact, risks may also come up after the project is delivered.