* Software Implementation :-

1. Programming style
2. Coding style
3. Selection of language and technology
4. Implementation means coding
5. There is so many challenges in implementation

1- Programming Style - Programming style also known as code style is a set of rules or guidelines used when writing the source code for a computer program. It is often claim that following a particular programming style will help programming read and understand source code confirming to the style and help to avoid introducing errors.

2- Coding style - Coding conventions are set of guidelines for a specific programming language that recommend programming style, practice and methods for each aspect of a program written comments, declaration, statements, white space, naming convention indentation, programming principles programming rules of thumb etc.
These are the guidelines for Software structure quality, software programmers are highly recommended to follow these

DATE : 9 / Mar / 2019

guidelines to help improving the readability of their source code make software maintenance easy.

* **Software implementation process issues and challenges :-**

The process of new software implementation requires specialist, checklist and quality assurance. Software change management tracking solutions and a robust cooperation with the vendor. We are going to cover some of the most important challenges that need to be taken into account when creating a software implementation plan. After listening challenges and issues we are then going to present some general tips and ideas that will help to make sure your next roll-out is successful.

1- Software legacy - One of the main issues, company need to take during the software implementation planning phase is system of software legacy. A solid change management strategy always takes into account all the limitations connected to legacy issues

and already addresses the problem by defining implementation rules that generate compatibility and software compliance. In some extreme cases, the company might need to consider going through a massive legacy system modernisation to partially or completely solve an issue.

2- Software certification — Another important aspect that cannot be neglect and which usually is specifically connected to the negotiation and due phase is making sure that the software meets all the certifications necessary for diplomat.
Software certification are important when it comes to both compliance and security.

3- IT infrastructure and integrations — Sometimes it's hard to integrate new services or tools within an existing infrastructure. Creating a company structure that mirrors a smooth customer experience however requires data to be centralize and analyze across different units to avoid siloed structures and whole in the customer journey. Imagine implementing marketing

automation software that integrate with the main interact with customer help storing solutions to triger action or cross reference data in order to implement a new technology.

4- Software training — One of the biggest risk connected to software implementation is the lack of proper software training. The project lead needs to set clear goals and deadlines. Sometimes a company wide software roll-out plan can be hindered by the lack of resource for software training. In this case organisation are often force to schedule different implementation mile stones across different units.

5- Software Portability and backward compatibility — There is nothing worse than creating a software implementation plan to roll-out a new platform or schedule a major company while upgrade to find out that new system is not working properly across different computing platform. This issues might lead to very high cost link to either software customization if possible or

hardware updates. At the same time we also want to make sure that new update guarantee for interoperability with older legacy system.

6- Changes in the team structure – Changes are often reflected in the team structure and roles and this also often holds true when it comes to implementing new software. A new software roll-out might need to be followed by small adjustments in roles and responsibilities. These can drive from the necessity of having an administrator or manager that work on the integration of the new solution or new reporting and data analyze figures within the team.

7- Data migration – Connected to the previous point data migration relies of backwards, downwards compatibility. This is essential to focus on the data migration very easily in the implementation process since this is a very time consuming and risk component of the implementation plan.

8- Security and stability – Every software implementation is a potential gate that

might lead to security branches rushing into a speedy roll-out might soon become an interact labyrinth of patches and work around that can jeopardize. The stability of the entire architecture. In some case, it is even recommended to carry-out vulnerability to verify that the structure is still robust and secure.

* Programming Support Environment :-

In computer program and software product development, the development environment is the set of process and programming tools used to create the program or software product. The term may sometime also employ the physical environment.

An integrated development environment is one in which the process and tools are coordinated to provide developers an orderly interface to add convenient view of the development process or at least the process of the writing code, testing it and packaging it for use.

An example of an IDE product is- Microsoft's visual studio .net, Net Beans, XCode, Web IDE's, edipse.

The term computer assisted software environment is generally used to describe a set of tools and practice that facilitate management of a software development project. Software environment is the term commonly used to refer to support an application. A software environment for a particular application could include the operating system, the data base system, specific development tool or compilers. development environment in which software coding is done. It include the technology use for front-end, back-end, computer configurations, memory disk space etc.

It also include the software development model following development tools, number of programs, the organisation policies or any kind of support needed in coding a software. production environment in which software is actually used by client. It comprise of operation system, computer configuration, browsers used by client, number of software users etc.

* Good Programming Style :-

Proper programming style signification reduce maintenance cost and increase the lifetime and functionality of software Most software disasters are rooted in poor style of programming. There are many best rules that lead to a better programming style.

1- Readability - Good code is written to be easily understand by colleagues. It is properly and consistently formatted and clear meaningful name for functions and variables, concise and accurate comments describe a natural decomposition of the software functionality into simple and specific functions. Any tricky sections are clearly noted. It should be easy to see why the program will work and reason that it should work in all cases.

2- Maintenability - Code should be written so that it is straight forward for another programmer to fix bugs for make change to its functionality later. Function should be general and assume as little as possible about preconditions.

DATE: / /

All important values should be robust to handle any possible input and produce a responsible result without crashing. Clear message should be output for input which is not allow.

3- Comments — Comments are the first step towards making computer program human readable. Comments should explain clearly everything about a program which is not obvious to a fair programmer.
The volume of comments written is meaningless quality is all that counts. Block comments are written using comments style. They should go at the top of every source file and generally include your name, the data, your code was written an overall description of the purpose of that program. Block comments should also precede most functions with a description of the functions purpose. These can be omitted for every obvious function only.

4- Indentation — Indentation is used to clearly make mark control flow in a program within any bracket block. All code is identical any one tap. These includes the

class body itself. Each additional for, while, if or switch structure introduce a new block which is indented even if brackets are omitted for one line statement any corresponding else statement should line-up.

5- White space — White space is meaningless to compiler but should be used consistently to improve readability. Typically three blank lines are left in between functions.

6- Output — A final overlook aspect of good programming style is how your program output results and information to users part of writing professional looking programs is providing clear instructions and results to the user of your program. This means proper English with no spelling error conditions. One must always assume that writing program to be use by somebody.

* Function uses :-

Function should be short and clear specific task. As much as possible they should be consider black box which do not depend on anything except their parameters

and can be handle any possible input gracefully. A common rule of thumb is the tag line rules. Usually functions longer than tag lines are trying to do much and should be simplified.

* **Maintenance :-**

Software maintenance in software engineering is the modification of a software product after delivery, to correct fault to improve information of other attributes. A common perception of maintenance is that it merely involve fixing defects. However one study indicates that our 80% of maintenance effort is use for non-corrective actions. Software resolution is the term used in software engineering. Software maintenance refers to the process of developing software initially than repeated updating its various reasons. Software maintenance is widely accepted part of SDLC nowadays.

It stand for all the modification and updation done after the delievery of software product. There are number of reasons why modification are required. Some of them are ~~be~~ briefly mentioned here -

1- Market condition – Policies which change over the time such as newly introduced constants like how to maintain.

2- Host modification – If any of the hardware and platform such as operating system of the target host change, software change are needed to keep adaptability.

3- Client requirement – Over the time, customer may ask for new features or functions in the software.

4- Organisation change – If there is any business level change at client such as reduction of organisation strength, acquiring another company, organisation venture into new business need to modify in the original software may arise.

- Types of maintenance :-

In a software lifetime, type of maintenance may vary based on its nature. it may be just a routine. Maintenance task as some bugs discover by some users or it

may be a large event in itself. Based on maintenance size or nature, following are some types of maintenance based on their characteristics.

1. Corrective maintenance – This include modification and updations done in order to correct of fix problems which are either discovered by some users or it may be a large event in itself based on maintenance. Order to correct fix problem or concluded by users error reports.

2. Adaptive maintenance – This include modification and updation applied to keep the software product up to date and turn to the everchanging world of technology and business environment.

3. Perfective maintenance – This includes modification and updation done in order to keep the software usable over long period of time. It include new features, new users requirement for refining the software and improve its reliability and performance.

DATE : / /

4. Preventive maintenance — This include modification and updation to prevent future problems of the software. It aims to attend problems which are not significant at this moment but may cause series issues in future.

* Cost of Maintenance :-
Report suggest that the cost of maintenance is high. A steady on estimate software maintenance found that the cost of maintenance is as high as 67% of the cost of entire software process cycle on an average. The cost of software maintenance is more than 50% of on SDLC phases. There are various factors which trigger maintenance cost so high.

- Real world factor affecting maintenance cost — The standard age of any software is considered upto 10 to 15 years older the software which were where main to work on slow machines with less memory and storage capacity cannot kept themselves changing against newly coming inhance softwares or modern hardware. As a technology advanced it become costly to

maintain old software. Most maintenance engineers are newbie and use trial and errors method to rectified problems.
Often changes made can easily hurt the original structure of the software. Making it hard for any subsequence changes.
Changes are often poorly undocumented which may cause more conflict in future.