UNIT-5 Software Maintenance and Software Project Management

Software as an Evolutionary Entity:

In software engineering, software evolution is referred to as the process of developing, maintaining, and updating software for various reasons.

It is a product and, at the same time, the vehicle for delivering a product. As a product, it delivers the computing potential embodied by computer hardware or, more broadly, a network of computers that are accessible by local hardware.

Need for Maintenance:

Need of maintenance occurs due to the following factors:

- Correct faults.
- Improve the design.
- Implement enhancements.
- Interface with other systems.
- Accommodate programs so that different hardware, software, system features, and telecommunications facilities can be used.
- Migrate legacy software.
- Retire software.

Purpose of Maintenance:

- Failure Avoidance
- Equipment Reliability
- Least Operating Costs
- Risk Reduction
- Maximum Production
- Defect Simulation

Categories of Maintenance:

1. **Corrective maintenance:**
   Corrective maintenance of a software product may be essential either to rectify some bugs observed while the system is in use, or to enhance the performance of the system.

2. **Adaptive maintenance:**
   This includes modifications and updations when the customers need the product to run on new platforms, on new operating systems, or when they need the product to interface with new hardware and software.

3. **Perfective maintenance:**
   A software product needs maintenance to support the new features that the users want or to change different types of functionalities of the system according to the customer demands.

4. **Preventive maintenance:**
   This type of maintenance includes modifications and updations to prevent future problems of the software. It goals to attend problems, which are not significant at this moment but may cause serious issues in future.

Cost of Maintenance:

Cost of maintenance includes all activities necessary for software to meet all its functional requirements throughout the life cycle.

The cost basically depends upon

1. Non-Technical factors
2. Technical factors

Non-Technical factors:
The Non-Technical factors include:
1. Application Domain
2. Staff stability
3. Program lifetime
4. Dependence on External Environment
5. Hardware stability

Technical factors:
Technical factors include the following:
1. Module independence
2. Programming language
3. Programming style
4. Program validation and testing
5. Documentation
6. Configuration management techniques

Efforts expanded on maintenance may be divided into productivity activities (for example analysis and evaluation, design and modification, coding).The following expression provides a module of maintenance efforts:
M= P+ K(C-D)

Where,

M: Total effort expanded on the maintenance. P: Productive effort.
K: An empirical constant.
C: A measure of complexity that can be attributed to a lack of good design and documentation.
D:A measure of the degree of familiarity with the software.

Software Re-Engineering:

- Software Reengineering is the process of updating software.
- Thisprocessincludesdevelopingadditionalfeaturesonthesoftwareandadding functionalities for better and more efficient software.
- This process also entails that the software product will have **improved quality and maintainability.**

Why is there a Need for Software Reengineering?

The need for software reengineering becomes an integral part of improving the quality of your products. This process adds more value to your business as it does not only better your services but also contributes added revenue.

How is Software Reengineering Done?

1. Reverse Engineering

- Software Reverse Engineering is a process of recovering the design, requirement specificationsandfunctionsofaproductfromananalysisofitscode.Itbuildsaprogram database and generates information from this.

- The purpose of reverse engineering is to facilitate the maintenance work by improving the understandability of a system and to produce the necessary documents for a legacy system.

Steps of Software Reverse Engineering:

1. **Collection Information:**
   This step focuses on collecting all possible information (i.e., source design documents etc.) about the software.

2. **Examining the information:**
   The information collected in step-1 as studied so as to get familiar with the system.

3. **Extracting the structure:**
This step concerns with identification of program structure in the form of structure chart where each node corresponds to some routine.

4. **Recording the functionality:**
During this step processing details of each module of the structure, charts are recorded using structured language like decision table, etc.

5. **Recording dataflow:**
From the information extracted in step-3 and step-4, set of data flow diagrams are derived to show the flow of data among the processes.

6. **Recording control flow:**
High level control structure of the software is recorded.

7. **Review extracted design:**
Design document extracted is reviewed several times to ensure consistency and correctness. It also ensures that the design represents the program.

8. **Generate documentation:**
Finally, in this step, the complete documentation including SRS, design document, history, overview, etc. are recorded for future use.

2. Restructuring

Once the reverse engineering is done and the appropriate specifications are identified, **restructuring** is performed. Restructuring deals with rearranging or reconstructing the source code and deciding whether to retain or change the programming conventions.

Another part of this procedure is the elimination or reconstruction of the parts of the source code that often cause errors in the software (may also be debugging).Aside from that, eliminating obsolete or older versions of certain parts of the system (such as programming implementation and hardware components) should keep the system updated.

3. Forward Engineering

The flow ends with forward engineering. This is the process of integrating the latest specifications based on the results of the evaluations from reverse engineering and restructuring.

In relation to the entirety of the process, this is defined relative to reverse engineering, where there is an effort to build backward, from a coded set to a model, or to break down the process of how software was integrated.

There is no specific SDLC model to follow in software reengineering. The model will always depend on what fits best with the environment and implementation of your product.

Software Configuration Management:

Software Configuration Management (SCM) is the task of tracking and controlling changes in the software.

Software configuration Management includes following activities

- Configurationidentification–Identifyingconfigurations,configurationitemsandbaselines
- Configuration control – Implementing a controlled change process. This is usually achieved by setting up a change control board whose primary function is to approve or reject all change requests that are sent against any baseline.
- Configuration status accounting – Recording and reporting all the necessary information on the status of the development process.
- Configuration auditing – Ensuring that configurations contain all their intended parts and are sound with respect to their specifying documents, including requirements, architectural specifications and user manuals.
- Build management–Managing the process and tools used for builds.
- Processmanagement–Ensuringadherencetotheorganization'sdevelopmentprocess.
- Environmentmanagement–Managingthesoftwareandhardwarethathostthesystem.
- Teamwork–Facilitate team interactions related to the process.
- Defecttracking–Makingsureeverydefecthastraceabilitybacktothesource.
- Reducedredundantwork–Thisprocessalsohelpustoreducetheredundantwork.
- Avoids configuration-related problems – Software configuration management process make sure that the configuration changes done in any environment which may include stage, UAT, Integration or products is being tracked and controlled.
- Facilitates team coordination – Software configuration management also enables team to co-ordinate at many scale to get done software development faster without compromising the quality of the software. Need for co-ordination between Dev, QA and End User.
- Helps in building management – managing tools used in builds and release which include SCM Server, Build Servers.
- More than one version of the software has to be supported – Software configuration management enables us to support following…

— Multiple Releases
— Custom Configured systems
— Systems under Development
— Control the cost involved in making changes to a system

Software Configuration Management Process:

The software configuration management process is a series of steps designed to track and manage all the defects, resources, codes, documents, hardware and budgets throughout a project.
SCM is an inter disciplinary process involving people at every level, including DevOps, developers, project managers/owners, SysAdmin and testers.

1. Planning and Identification

The first step in the process is planning and identification. In this step, the goal is to plan for the development of the software project and identify the items within the scope. This is accomplished by having meetings and brainstorming sessions with your team to figure out the basic criteria for the rest of the project.
Part of this process involves figuring out how the project will proceed and identifying the exit criteria. This way your team will know how to recognize when all of the goals of the project have been met.

Specific activities during this step include:

- Identifying items like test cases, specific action requirements, and code modules
- Identifying each computer software configuration item in the process
- Group basic details of why, when, and what changes will be made and who will be in charge of making them
- Create a list of necessary resources, like tools, files, documents, etc.

2. Version Control and Baseline

The version control and baseline step ensures the continuous integrity of the product by identifying an accepted version of the software. This baseline of design at a specific time in the SCM process and can only be altered through a formal procedure.
The point of this step is to control the changes being made to the product. As the project develops, new baselines are established, resulting in several versions of the software.

This step involves the following activities:

- Identifying and classifying the components that are covered by the project
- Developing a way to track the hierarchy of different versions of the software
- Identifying the essential relationships between various components
- Establishingvariousbaselinesfortheproduct,includingdevelopmental,functional,and product baselines
- Developing a standardized label scheme for all products, revisions, and files so that everyone is on the same page.

Baselining a project attribute forces formal configuration change control processes to be enacted in the event that these attributes are changed.

3. Change Control:

Change control is the method used to ensure that any changes that are made are consistent with the rest of the project. Having these controls in place helps with quality assurance, and the approval and release of new baseline. Change control is essential to the successful completion of the project.
In this step, requests to change configurations are submitted to the team and approved or denied by the software configuration manager. The most common types of requests are to add or edit various configuration items or change user permissions.

This procedure includes:

- Controlling ad-hoc changes requested by the client
- Checking the merit of the change request by examining the overall impact they will have on the project
- Making approved changes or explaining why change requests were denied.

4. Configuration Status Accounting

The next step is to ensure the project is developing according to the plan by testing and verifyingaccordingtothepredeterminedbaselines.Itinvolveslookingatreleasenotesand related documents to ensure the software meets all functional requirements.
Configuration status accounting tracks each version released during the process, assessing what is new in each version and why the changes were necessary. Some of the activities in this step include:

- Recording and evaluating changes made from one baseline to the next
- Monitoring the status and resolution of fall change requests
- Maintaining documentation of each change made as a result of change requests and to reach another baseline
- Checking previous versions for analysis and testing.

5. Audits and Reviews

The final step is a technical review of every stage in the software development lifecycle. Audits and reviews look at the process, configurations, workflow, change requests, and everything that has gone into developing each baseline throughout the project's development.
The team performs multiple reviews of the application to verify its integrity and also put together essential accompanying documentation such as release notes, user manuals, and installation guides.

Activities in this step include:

- Making sure that the goals laid out in the planning and identification step are met
- Ensuring that the software complies with identified configuration control standards
- Making sure changes from baselines match the reports
- Validating that the project is consistent and complete according to the goals of the project.

Who is involved in the software configuration process?

The SCM process is multidisciplinary, involving just about every member of the software development team.

1. **Configuration Manager:** The configuration manager is in charge of determining who is responsible for what throughout the development process. They make sure everyone follows the SCM process across the project and have the final say on all change requests.
2. **Project Manager**

The project manager's role is integral to the SCM process. They set the timeframe of the project to ensure that it meets completion deadlines and they also generate reports about the

team's progress. Another important role of the project manager is to ensure that every member of the team is following the predetermined guidelines for creating, making changes to, and testing the software.

3. **Software Developers**

Developers are responsible for writing code during development as well as accommodating any approved change requests.

4. **Auditor**

This role is in charge of all audits and reviews and must make sure that the final release is complete and consistent.

What are the advantages of using configuration management tools?

There are a number of tools available to help facilitate the software configuration management process. The purpose of these tools is the automation of traditionally manual tasks, allowing for greater accuracy, speed and control. More specifically, they can help with:

- **Alerts and Reports:** A good SCM tool will provide alerts and reports if there are any deviations from the agreed upon baseline. This data will be pushed through in close to real-time, allowing managers to act fast if something goes off track.
- **Track Changes:** SCM tools will automatically track changes to servers or applications and will also allow manual entry of such data. Change auditing can also be done via monitoring script outputs.
- **Configuration Comparisons:** The best software configuration management tools will provide a way to identify differences between configurations.
- **Faster Troubleshooting:** Errors, missteps, and issues are identified quickly so that developers can take action before the problem grows.
- **Inventory Tracking:** Most SCM tools will feature a way to track hardware and software assets so that you don't have to keep a manual list.
- **Patch Management:** SCM tools can help you track all the details surrounding patch management as you distribute updated software.

Are there any downsides (Disadvantages) to using a software configuration management tool?

There are something's to consider before embracing an SCM tool, including:

- **Resource Drain:** You must have the resources to support the process from beginning to end
- **KnowledgeLimitations:**Everyoneinvolvedmusthaveaprofoundknowledgeofthe software management tools being used
- **SMB Disadvantage:** The scope of what is needed to use these tools effectively maybe difficult for a small business to support
- **Hardware Specs:** Fast and highly configured hardware is required for the process to run smoothly.

CASE Tools:

The set of application programs to automate software development life cycle activities and are used by managers in a project, engineers and analysts to build a software system is called CASE tools and the software development cycle stages can be simplified using several tools such as design, analysis, project management, database management, documentation, etc. and the use of these tools speeds up the project development to obtain desired results.

Components of CASE Tools

There are several components are based on their usage in different stages of the software development life cycle. They are:

- **Central Repository:** A central repository is required by the tools to serve as a common source of integrated and consistent information. The central place of storage consisting of specifications of product, documents requirement, diagrams and reports and information about the management is a central repository. The central repository also acts as a data dictionary.
- **Upper:** Planning, analysis, and designing of different stages of the software development life cycle can be performed using upper case.
- **Lower:** Implementation, testing, and maintenance can be performed using lower case.
- **Integrated:** All the stages of the software development life cycle right from the gathering of requirements for testing and documentation can be performed using integrated tools.

Types of CASE Tools

There are several types of tools available.

1. Diagram Tools

The components of the system, data flow, control flow among the various components of software and the structure of the system can be represented in graphical form using diagram tools.

Example: The state-of-the-art flow charts can be created using flow chart maker tool.

2. Process Modeling

The software process model can be created using process modeling tools for software development. The managers can choose a process model using process modeling tools or make modifications depending upon the software product requirements.

3. Project Management

Planning of the project, estimation of cost and efforts, scheduling of project and planning of resources can be done using project management tools. All the steps in the execution of the project must be strictly followed by the managers in management of software project. The project information can be stored and shared in real time using the tools of project management throughout the organization.

Examples: Creative Pro Office, trac project, Basecamp etc.

4. Documentation Tools

Before the beginning of software process, documentation of the software project must begin. This documentation must cover the all the software development life cycle phases and the completion of the software development phase as well. The documents are generated by the documentation tools for both technical and end users. The in-house professionals in the development team who refer the manual maintained for the system, manual maintained for reference, manual for training, manuals for installation etc. make the technical users. The functioning of the system and how system works is described in the end user documents.

Example: Doxygen, adobe robohelp, DrExplainetc.

5. Analysis

Requirements gathering, inconsistency checks, diagrams inaccuracy, redundancies in the data etc. can be checked using analysis.

Example: For requirement analysis are Accept 360, Accompa, casecomplete etc. Total analysis can be done using visible analyst.

6. Design

The block structure of the software can be designed by the software designers using design tools which are again broken down into smaller modules using techniques of refinement. The detailing of every module and the interconnections between the modules can be done using this.

Example: Animated software design.

7. Configuration Management Tools

Whenever one version of software instance is released, configuration management tools deals with the following:

- Management of revision and version
- Configuration management of baseline
- Change control management

Automatic tracking, management of version, and management of release can be done with the help of configuration management.

Example: Git, AccuRevetc.

8. Change Control

Change Control area part of configuration management. The changes that occur in the software after fixing its baseline or after the first release of the software are dealt by change control tools. Tracking the changes, management of files, management of codes etc. can be automated

using change control. The change policy of the organization can be enforced by using change control.

9. Programming

The programming environments like integrated development environment, library consisting of in built modules, simulation are all included in programming tools. The development of software product is aided by these and simulation and testing features are included.

Example: C scope for searching code in C, Eclipse.

10. Prototyping

The simulated version of the software product to be built is called a prototype in software. The look and feel of the product is provided by the prototype and several aspects of the actual product can be simulated using prototyping. Graphical libraries are contained in the prototyping tools. User interfaces and design that are hardware independent can be created using prototyping. Rapid prototype scan be built using prototyping based on the existing information. The software prototype can be simulated using prototyping tools.

Example: Mockup builder, Serena prototype composer etc.

11. Web Development

The web pages like forms, text, script, graphic etc. can be designed using web development tools. The web page that is being developed can be previewed to see how it looks after completion using web development.

Example: Adobe Edge Inspect, Foundation3, brackets etc.

12. Quality Assurance

Monitoring the engineering process and methods used for software development to ensure the quality is as per the standards of the organization can be performed using quality assurance tools. The configuration change control and software testing tools come under the category of QA tools.

Example: Soap Test, Jmeter, AppsWatch, etc.

13. Maintenance

If there are any modifications after the delivery of the software product can be done through software maintenance tools. Techniques for automatically logging, error reporting, generation of error tickets automatically and root cause analysis are used in the maintenance phase of the software development life cycle to help the software organizations.

Example: Bugzilla for tracking defects etc.

Cocomo (Constructive Cost Model):

Cocomo (Constructive Cost Model) is a regression model based on LOC, i.e **number of Lines of Code**. It is a procedural cost estimate model for software projects and is often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time, and quality. It was proposed by Barry Boehm in 1981 and is based on the study of 63 projects, which makes it one of the best-documented models.

The key parameters which define the quality of any software products, which are also an outcome of the Cocomo are primarily Effort & Schedule:

- **Effort:** Amount of labor that will be required to complete a task. It is measured in person-months units.
- **Schedule:** Simply means the amount of time required for the completion of the job, which is, of course, proportional to the effort put in. It is measured in the units of time such as weeks, months.

Different models of Cocomo have been proposed to predict the cost estimation at different levels, based on the amount of accuracy and correctness required. All of these models can be applied to a variety of projects, whose characteristics determine the value of constant to be used in subsequent calculations. These characteristics pertaining to different system types are mentioned below.

Boehm's definition of organic, semidetached, and embedded systems:

1. **Organic** – A software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved in the past and also the team members have a nominal experience regarding the problem.
2. **Semi-detached** – A software project is said to be a Semi-detached type if the vital characteristics such as team size, experience, knowledge of the various programming environment lie in between that of organic and embedded. The projects classified as Semi-Detached are comparatively less familiar and difficult to develop compared to the organic ones and require more experience and better guidance and creativity. E.g. Compilers or different Embedded Systems can be considered of Semi-Detached type.
3. **Embedded** – A software project requiring the highest level of complexity, creativity, and experience requirement fall under this category. Such software requires a larger team size than the other two models and also the developers need to be sufficiently experienced and creative to develop such complex models.

All the above system types utilize different values of the constants used in Effort Calculations.

Types of Models: COCOMO consists of a hierarchy of three increasingly detailed and accurate forms. Any of the three forms can be adopted according to our requirements. These are types of COCOMO model:

1. Basic COCOMO Model
2. Intermediate COCOMO Model
3. Detailed COCOMO Model

The first level, **Basic COCOMO** can be used for quick and slightly rough calculations of Software Costs. Its accuracy is somewhat restricted due to the absence of sufficient factor considerations.

Intermediate COCOMO takes these Cost Drivers into account and **Detailed COCOMO** additionally accounts for the influence of individual project phases, i.e in case of Detailed it accounts for both these cost drivers and also calculations are performed phase-wise henceforth producing a more accurate result. These two models are further discussed below.

Estimation of Effort: Calculations –

Basic Model –

E=ax(KLOC)b

D=cx(Effort)d P

= effort/time

Where,

E is effort applied in person-months.

D is development time in months.

P is the total no. of persons required to accomplish the project.

The constant values a, b, c, and d for the Basic Model for the different categories of the system

The above formula is used for the cost estimation of for the basic COCOMO model, and also is used in the subsequent models. The constant values a, b, c and d for the Basic Model for the different categories of system:

Software Projects	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

The effort is measured in Person-Months and as evident from the formula is dependent on Kilo-Lines of code.
The development time is measured in months.

These formulas are used as such in the Basic Model calculations, as not much consideration of different factors such as reliability, expertise is taken into account, henceforth the estimate is rough.

Output:
The mode is Organic

Effort = 10.289 Person-Month

Development Time = 6.06237 Months

Average Staff Required = 2 Persons

4. **Intermediate Model–**
The basic Cocomo model assumes that the effort is only a function of the number of lines of code and some constants evaluated according to the different software systems. However, in reality, no system's effort and schedule can be solely calculated on the basis of Lines of Code. For that, various other factors such as reliability, experience, Capability. These factors are known as Cost Drivers and the Intermediate Model utilizes 15 such drivers for cost estimation.

Classification of Cost Drivers and their attributes:

(i) Product attributes–
- Required software reliability extent
- Size of the application database
- The complexity of the product

(ii) Hardware attributes–
- Run-time performance constraints
- Memory constraints
- The volatility of the virtual machine environment
- Required turn about time

(iii) Personnel attributes–
- Analyst capability
- Software engineering capability
- Applications experience
- Virtual machine experience

- Programming language experience

(iv) Project attributes–

- Use of software tools
- Application of software engineering methods
- Required development schedule

Cost Drivers	Very Low	Low	Nominal	High	Very High
Product Attributes					
Required Software Reliability	0.75	0.88	1.00	1.15	1.40
Size of Application Database		0.94	1.00	1.08	1.16
Complexity of The Product	0.70	0.85	1.00	1.15	1.30
Hardware Attributes					
Runtime Performance Constraints			1.00	1.11	1.30
Memory Constraints			1.00	1.06	1.21
Volatility of the virtual machine environment		0.87	1.00	1.15	1.30
Required turn about time		0.94	1.00	1.07	1.15
Personnel attributes					
Analyst capability	1.46	1.19	1.00	0.86	0.71
Applications experience	1.29	1.13	1.00	0.91	0.82
Software engineer capability	1.42	1.17	1.00	0.86	0.70
Virtual machine experience	1.21	1.10	1.00	0.90	
Programming language experience	1.14	1.07	1.00	0.95	
Project Attributes					

Cost Drivers	Very Low	Low	Nominal	High	Very High
Application of software engineering methods	1.24	1.10	1.00	0.91	0.82
Use of software tools	1.24	1.10	1.00	0.91	0.83
Required development schedule	1.23	1.08	1.00	1.04	1.10

The project manager is to rate these 15 different parameters for a particular project on a scale of one to three. Then, depending on these ratings, appropriate cost driver values are taken from the above table. These 15 values are then multiplied to calculate the EAF (Effort Adjustment Factor). The Intermediate COCOMO formula now takes the form:

$$E = (a(KLOC)^b) * EAF$$

The values of a and b in case of the intermediate models are as follows:

Software Projects	a	b
Organic	3.2	1.05
Semi Detached	3.0	1.12
Embedded	2.8	1.20

5. **Detailed Model** –
Detailed COCOMO incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step of the software engineering process. The detailed model uses different effort multipliers for each cost driver attribute. In detailed cocomo, the whole software is divided into different modules and then we apply COCOMO indifferent modules to estimate effort and then sum the effort.
The Six phases of detailed COCOMO are:

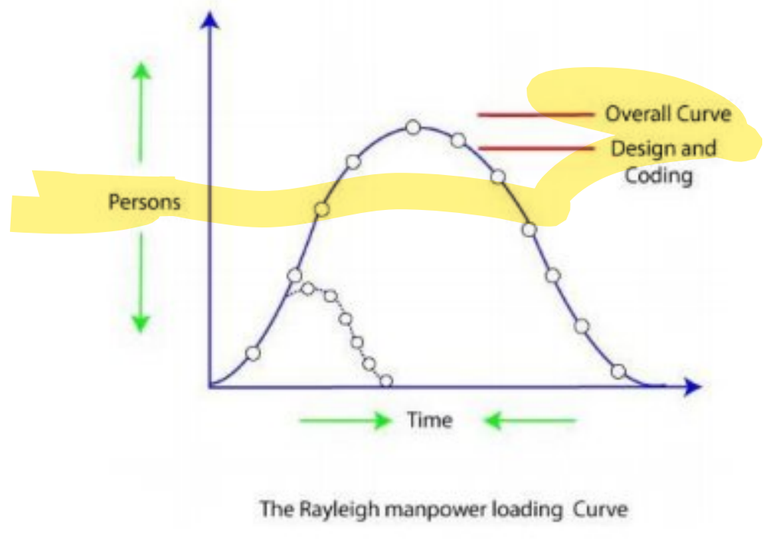
1. Planning and requirements
2. System design
3. Detailed design
4. Module code and test
5. Integration and test
6. Cost Constructive model

The effort is calculated as a function of program size and a set of cost drivers are given according to each phase of the software lifecycle.

Resource Allocation Models:

Putnam Resource Allocation Model:

The Lawrence Putnam model describes the time and effort requires finishing a software project of a specifiedsize.Putnammakesauseofaso-calledTheNorden/RayleighCurvetoestimateprojecteffort, schedule & defect rate as shown in fig:



The Rayleigh manpower loading Curve

Putnam noticed that software staffing profiles followed the well known Rayleigh distribution. Putnam used his observation about productivity levels to derive the software equation:

$$L = C_k K^{1/3} t_d^{4/3}$$

The various terms of these expression areas follows:

K is the total effort expended (in PM) in product development, and L is the product estimate in **KLOC** .

t_d correlate to the time of system and integration testing. Therefore, t_d can be relatively considered as the time required for developing the product.

C_k Is the state of technology constant and reflects requirements that impede the development of the program.

Typical values of C_k=2 for poor development environment

C_k=8 for good software development environment

C_k=11for an excellent environment (in addition to following software engineering principles, automated tools and techniques are used).

The exact value of C_k for a specific task can be computed from the historical data of the organization developing it.

Putnam proposed that optimal staff develop on a project should follow the Rayleigh curve. Only a small number of engineers are required at the beginning of a plan to carryout planning and specification tasks. As the project progresses and more detailed work are necessary, the number of engineers reaches a peak. After implementation and unit testing, the number of project staff falls.

Effect of a Schedule change on Cost

Putnam derived the following expression:

$$L = C_k K^{1/3} t_d^{4/3}$$

Where, **K** is the total effort expended (in PM) in the product development

List the product size in KLOC

T_d corresponds to the time of system and integration testing

C_k Is the state of technology constant and reflects constraints that impede the progress of the program

Now by using the above expression, it is obtained that,

$$K = L^3 / C_k^3 t_d^4$$

Or $$K = C / t_d^4$$

For the same product size, $C=L^3/C^3$

Or $$\frac{K_1}{K_2} = t_{d2}^4 / t_{d1}^4$$

Or $$K \propto 1/t_d^4$$

Or, $$\text{cost} \propto 1/t_d$$

(As project development effort is equally proportional to project development cost)

From the above expression, it can be easily observed that when the schedule of a project is compressed, the required development effort as well as project development cost increases in proportion to the fourth power of the degree of compression. It means that a relatively small compression in delivery schedule can result in a substantial penalty of human effort as well as development cost.

Software Risk Analysis and Management:

What is Risk Analysis?

Risk Analysis in project management is a sequence of processes to identify the factors that may affect a project's success. These processes include risk identification, analysis of risks, risk management and control, etc. Proper risk analysis helps to control possible future events that may harm the overall project. It is more of a pro-active than a reactive process.

How to Manage Risk?

RiskManagementinSoftwareEngineeringprimarilyinvolvesfollowingactivities:

Plan risk management

It is the procedure of defining how to perform risk management activities for a project.

Risk Identification

It is the procedure of determining which risk may affect the project most. This process involves documentation of existing risks.

The input for identifying risk will be

- Risk management plan
- Project scope statement
- Cost management plan
- Schedule management plan
- Human resource management plan
- Scope baseline
- Activity cost estimates
- Activity duration estimates
- Stakeholder register
- Project documents
- Procurement documents
- Communication management plan
- Enterprise environmental factor
- Organizational process assets
- Perform qualitative risk analysis
- Perform quantitative risk analysis
- Plan risk responses
- Monitor and control risks

The output of the process will be a

- Risk register

Perform qualitative risk analysis

It is the process of prioritizing risks for further analysis of project risk or action by combining and assessing their probability of occurrence and impact. It helps managers to lessen the uncertainty level and concentrate on high priority risks.

Plan risk management should take place early in the project; it can impact on various aspects for example: cost, time, scope, quality and procurement.

The inputs for qualitative Project Risk Analysis and Management includes

- Risk management plan

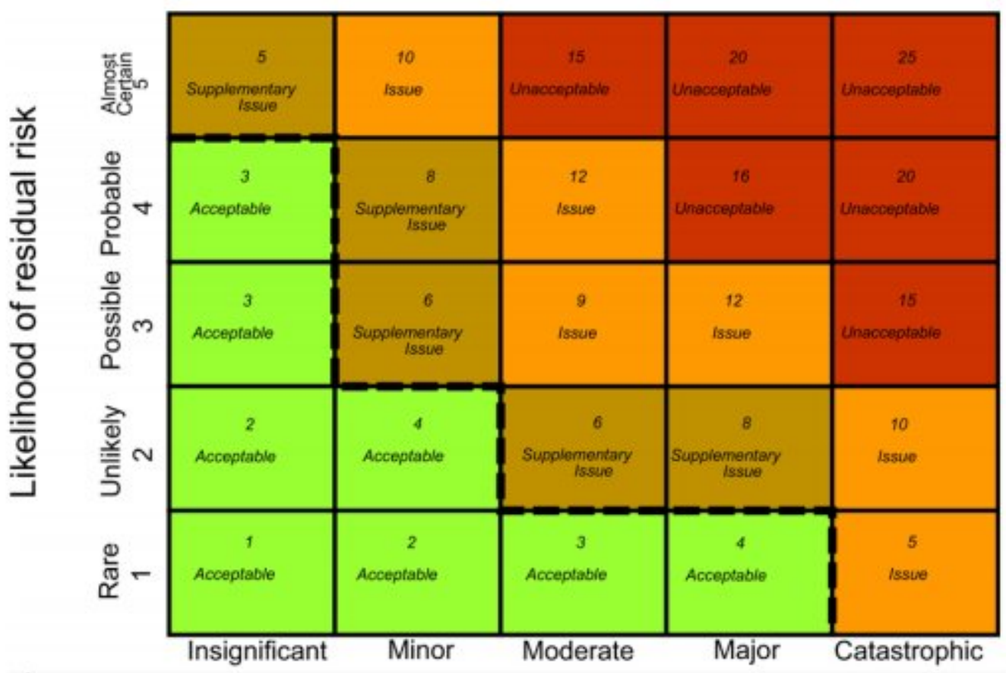
- Scope baseline
- Risk register
- Enterprise environmental factors
- Organizational process assets

The output of this stage would be

- Project documents updates

Quantitative risk analysis

It is the procedure of numerically analyzing the effect of identified risks on overall project objectives. In order to minimize the project uncertainty, this kind of analysis are quite helpful for decision making.



Risk Management Matrix

The input of this stage is

- Risk management plan
- Cost management plan
- Schedule management plan
- Risk register
- Enterprise environmental factors
- Organizational process assets

While the output will be

- Project documents updates

Plan risk responses

To enhance opportunities and to minimize threats to project objectives plan risk response is helpful. It addresses the risks by their priority, activities into the budget, schedule, and project management plan.

The inputs for plan risk responses are

- Risk management plan
- Risk register

While the output are

- Project management plan updates
- Project documents updates

Control Risks

Control risk is the procedure of tracking identified risks, identifying new risks, monitoring residual risks and evaluating risk.

The inputs for this stage includes

- Software Project management plan
- Risk register
- Work performance data
- Work performance reports

The output of this stage would be

- Work performance information
- Change requests
- Project management plan updates
- Project documents updates
- Organizational process assets updates

Project Procurement Management

Project Procurement Management includes the processes of purchasing or acquiring products needed to run a business. The organization can be a seller, buyer or service provider.

ProjectProcurementManagementalsoincludescontrollinganycontractissuedbyanoutside organization and get work done outside the project team.

Plan Procurement Management includes four stages like

- Plan Procurement Management
- Conduct Procurements
- Control Procurements
- Close Procurements

The input in the plan procurement management is

- Requirements documentation
- Teaming agreements
- Risk register
- Scope baseline

- Project schedule
- Activity cost estimates
- Cost performance baseline
- Risk related contract decisions
- Enterprise environmental factors
- Organizational process assets

Conduct Procurement process

Conduct Procurement process involves activities like

- Selecting a seller
- Receiving seller responses
- Awarding a contract

The benefit of conducting procurement process is that it provides alignment of external and internal stakeholder expectations through established agreements.

The input of the conduct procurement process includes

- Project management plan
- Documents for procurement
- Sources election criteria
- Qualified seller list
- Seller proposals
- Project documents
- Make or buy decisions
- Teaming agreements
- Organizational process assets

Control Procurements

It is the process of monitoring contract performance and correction to the contract as per the guidelines. It will ensure that buyers and sellers both meet the procurement requirement according to the terms of the legal agreement.

The input of the Control Procurements include

- Project management plan
- Procurement documents
- Agreements
- Approved change requests
- Work performance reports
- Work performance data

The output includes

- Work performance information
- Change requests
- Project management plan updates

- Project documents updates
- Organizational process assets updates

Close procurements

This step involves documenting agreements and other documents for future reference.

The input of this tool includes

- Project management plan
- Procurement documents

The output of this tool includes

- Closed procurements
- Organizational process assets updates

Manage Stakeholder Engagement

A stakeholder is an integral part of any project; their decision can leave a deep impact on project deliverables. In this process, the first part is to identify people, groups or organizations that could impact on the project while the second part is to analyze stakeholder expectations.

It also focuses on continuous communication with stakeholders to understand their needs and expectations.

Identifying Stakeholders

It is the process of identifying the groups, people or organization that can influence project outcomes. It allows the project manager to identify appropriate stakeholders.

Plan Stakeholder Management

It is the process of preparing a strategy to involve stakeholders throughout the project life cycle. It defines clear, actionable plan to interact with project Stakeholders.

The input for Plan Stakeholder Management includes

- Project management plan
- Stakeholder register
- Enterprise environmental factors
- Organizational process assets

The output of this

- Stakeholder management plan
- Project documents updates

Manage Stakeholder Engagement

In this stage, stakeholders are communicated to understand their expectations, address issues and foster appropriate stakeholder engagement in project activities. It allows the project manager to achieve project success without conflicting with stakeholder's decision.

The input of this stage is

- Stakeholder management plan
- Communication management plan
- Change log
- Organization process assets

While the output of this stage is

- Issue log
- Change request
- Project management plan updates
- Project documents updates
- Organizational process assets updates

Control Stakeholder Engagement

It is the process of monitoring stakeholder engagement in the project and adjusting strategies as per requirements. It will increase the stakeholder engagement activities as the project evolves and progresses.

The input for this stage include

- Project management plan
- Issue log
- Work performance data
- Project documents

The output of this stage include

- Work performance information
- Change requests
- Project management plan updates
- Project documents updates
- Organizational process assets updates