SOFTWARE ENGINEERING & PROJECT MANAGEMENT

Degree of Rigor

Even for a project of a particular type, the degree of rigor with which the software process is applied may vary significantly. The degree of rigor is a function of many project characteristics. As an example, small, non-mission critical projects can generally be addressed with somewhat less rigor than large, complex mission critical applications. It should be noted, however, that all projects must be conducted in a manner that results in timely, high quality deliverables. Four different degrees of rigor are defined for the APM:

1. Casual. All APM framework activities are applied, but only a minimum task set is required. In general, umbrella tasks will be minimized and documentation requirements will be reduced. All basic principles of software engineering are still applicable.
2. Structured. The APM framework will be applied for this project. Framework activities and related tasks appropriate to the project type will be applied and umbrella activities necessary to ensure high quality will be applied. SQA, SCM, documentation and measurement tasks will be conducted in a streamlined manner.
3. Strict. The APM will be applied for this project with a degree of discipline that will ensure high quality. All umbrella activities will be applied and robust documentation will be produced.
4. Quick Reaction. The APM will be applied for this project, but because of an emergency situation, only those tasks essential to maintaining good quality will be applied. "Back-filling" (e.g., developing a complete set of documentation, conducting additional reviews) will be accomplished after the application/product is delivered to the customer.

FORMAL TECHNICAL REVIEWS

A formal technical review is a software quality assurance activity performed by software engineers (and others). The objectives of the FTR are

1. To uncover errors in function, logic, or implementation for any representation of the software;
2. To verify that the software under review meets its requirements;
3. To ensure that the software has been represented according to predefined standards;
4. To achieve software that is developed in a uniform manner; and
5. To make projects more manageable.

In addition, the FTR serves as a training ground, enabling junior engineers to observe different approaches to software analysis, design, and implementation. The FTR also serves to promote backup and continuity because a number of people become familiar with parts of the software that they may not have otherwise seen.

The FTR is actually a class of reviews that includes walkthroughs, inspections, round-robin reviews and other small group technical assessments of software. Each FTR is conducted as a meeting and will be successful only if it is properly planned, controlled, and attended. In the sections that follow, guidelines similar to those for a walk through are presented as a representative formal technical review.

The Review Meeting

Regardless of the FTR format that is chosen, every review meeting should abide by the following constraints:

- Between three and five people (typically) should be involved in the review.
- Advance preparation should occur but should require no more than two hours of work for each person.
- The duration of the review meeting should be less than two hours.

Given these constraints, it should be obvious that an FTR focuses on a specific (and small) part of the overall software. For example, rather than attempting to review an entire design, walkthroughs are conducted for each component or small group of components. By narrowing focus, the FTR has a higher likelihood of uncovering errors.

The focus of the FTR is on a work product (e.g., a portion of a requirements specification, a detailed component design, a source code listing for a

component). The individual who has developed the work product—the producer—informs the project leader that the work product is complete and that a review is required. The project leader contacts a review leader, who evaluates the product for readiness, generates copies of product materials, and distributes them to two or three reviewers for advance preparation. Each reviewer is expected to spend between one and two hours reviewing the product, making notes, and otherwise becoming familiar with the work. Concurrently, the review leader also reviews the product and establishes an agenda for the review meeting, which is typically scheduled for the next day.

The review meeting is attended by the review leader, all reviewers, and the producer. One of the reviewers takes on the role of the recorder; that is, the individual who records (in writing) all important issues raised during the review. The FTR begins with an introduction of the agenda and a brief introduction by the producer. The producer then proceeds to "walk through" the work product, explaining the material, while reviewers raise issues based on their advance preparation. When valid problems or errors are discovered, the recorder notes each.

At the end of the review, all attendees of the FTR must decide whether to (1) accept the product without further modification, (2) reject the product due to severe errors (once corrected, another review must be performed), or (3) accept the product provisionally (minor errors have been encountered and must be corrected, but no additional review will be required). The decision made, all FTR attendees complete a sign-off, indicating their participation in the review and their concurrence with the review team's findings.

Review Reporting and Record Keeping

During the FTR, a reviewer (the recorder) actively records all issues that have been raised. These are summarized at the end of the review meeting and a review issues list is produced. In addition, a formal technical review summary report is completed.

A review summary report answers three questions:
1. What was reviewed?
2. Who reviewed it?
3. What were the findings and conclusions?

The review summary report is a single page form (with possible attachments). It becomes part of the project historical record and may be distributed to the project leader and other interested parties.

The review issues list serves two purposes:
(1) To identify problem areas within the product and
(2) To serve as an action item checklist that guides the producer as corrections are made. An issues list is normally attached to the summary report.

It is important to establish a follow-up procedure to ensure that items on the issues list have been properly corrected. Unless this is done, it is possible that issues raised can “fall between the cracks.” One approach is to assign the responsibility for followup to the review leader.

Review Guidelines

Guidelines for the conduct of formal technical reviews must be established in advance, distributed to all reviewers, agreed upon, and then followed. A review that is uncontrolled can often be worse that no review at all. The following represents a minimum set of guidelines for formal technical reviews:

1. Review the product, not the producer. An FTR involves people and egos. Conducted properly, the FTR should leave all participants with a warm feeling of accomplishment. Conducted improperly, the FTR can take on the aura of an inquisition. Errors should be pointed out gently; the tone of the meeting should be loose and constructive; the intent should not be to embarrass or belittle. The review leader should conduct the review meeting to ensure that the proper tone and attitude are maintained and should immediately halt a review that has gotten out of control.

2. Set an agenda and maintain it. One of the key maladies of meetings of all types is drift. An FTR must be kept on track and on schedule. The review leader is chartered with the responsibility for maintaining the meeting schedule and should not be afraid to nudge people when drift sets in.

3. Limit debate and rebuttal. When an issue is raised by a reviewer, there may not be universal agreement on its impact. Rather than spending time debating the question, the issue should be recorded for further discussion off-line.

4. Focus on problem areas, but don’t attempt to solve every problem noted. A review is not a problem-solving session. The solution of a problem can often be accomplished by the producer alone or with the help of only one other individual. Problem solving should be postponed until after the review meeting.

5. Take written notes. It is sometimes a good idea for the recorder to make notes on a wall board, so that wording and priorities can be assessed by other reviewers as information is recorded.

6. Limit the number of participants and insist upon advance preparation. Two heads are better than one, but 14 are not necessarily better than 4. Keep the number of people involved to the necessary minimum. However, all review team members must prepare in advance. Written comments should be solicited by the review leader (providing an indication that the reviewer has reviewed the material).

7. Develop a checklist for each product that is likely to be reviewed. A checklist helps the review leader to structure the FTR meeting and helps each reviewer to focus on important issues. Checklists should be developed for analysis, design, code, and even test documents.

8. Allocate resources and schedule time for FTRs. For reviews to be effective, they should be scheduled as a task during the software engineering process. In addition, time should be scheduled for the inevitable modifications that will occur as the result of an FTR.

9. Conduct meaningful training for all reviewers. To be effective all review participants should receive some formal training. The training should stress both process-related issues and the human psychological side of reviews. Freedman and Weinberg estimate a one-month learning curve for every 20 people who are to participate effectively in reviews.

10. Review your early reviews. Debriefing can be beneficial in uncovering problems with the review process itself. The very first product to be reviewed should be the review guidelines themselves.
Because many variables (e.g., number of participants, type of work products, timing and length, specific review approach) have an impact on a successful review, a software organization should experiment to determine what approach works best in a local context. Porter and his colleagues provide excellent guidance for this type of experimentation.

ISO 9001 certification:

A summary of the main requirements of ISO 9001 as they relate of software development is as follows. Section numbers in brackets correspond to those in the standard itself:

Management Responsibility (4.1)

- The management must have an effective quality policy.
- The responsibility and authority of all those whose work affects quality must be defined and documented.
- A management representative, independent of the development process, must be responsible for the quality system. This requirement probably has been put down so that the person responsible for the quality system can work in an unbiased manner.
- The effectiveness of the quality system must be periodically reviewed by audits.

Quality System (4.2)

A quality system must be maintained and documented.

Contract Reviews (4.3)

Before entering into a contract, an organization must review the contract to ensure that it is understood, and that the organization has the necessary capability for carrying out its obligations.

Design Control (4.4)

- The design process must be properly controlled, this includes controlling coding also. This requirement means that a good configuration control system must be in place.
- Design inputs must be verified as adequate.
- Design must be verified.
- Design output must be of required quality.
- Design changes must be controlled.

Document Control (4.5)

- There must be proper procedures for document approval, issue and removal.
- Document changes must be controlled. Thus, use of some configuration management tools is necessary.

Purchasing (4.6)

Purchasing material, including bought-in software must be checked for conforming to requirements.

Purchaser Supplied Product (4.7)

Material supplied by a purchaser, for example, client-provided software must be properly managed and checked.

Product Identification (4.8)

The product must be identifiable at all stages of the process. In software terms this means configuration management.

Process Control (4.9)

- The development must be properly managed.
- Quality requirement must be identified in a quality plan.

Inspection and Testing (4.10)

In software terms this requires effective testing i.e., unit testing, integration testing and system testing. Test records must be maintained.

Inspection, Measuring and Test Equipment (4.11)

If integration, measuring, and test equipments are used, they must be properly maintained and calibrated.

Inspection and Test Status (4.12)

The status of an item must be identified. In software terms this implies configuration management and release control.

Control of Nonconforming Product (4.13)

In software terms, this means keeping untested or faulty software out of the released product, or other places whether it might cause damage.

Corrective Action (4.14)

This requirement is both about correcting errors when found, and also investigating why the errors occurred and improving the process to prevent occurrences. If an error occurs despite the quality system, the system needs improvement.

Handling, (4.15)

This clause deals with the storage, packing, and delivery of the software product.

Quality records (4.16)

Recording the steps taken to control the quality of the process is essential in order to be able to confirm that they have actually taken place.

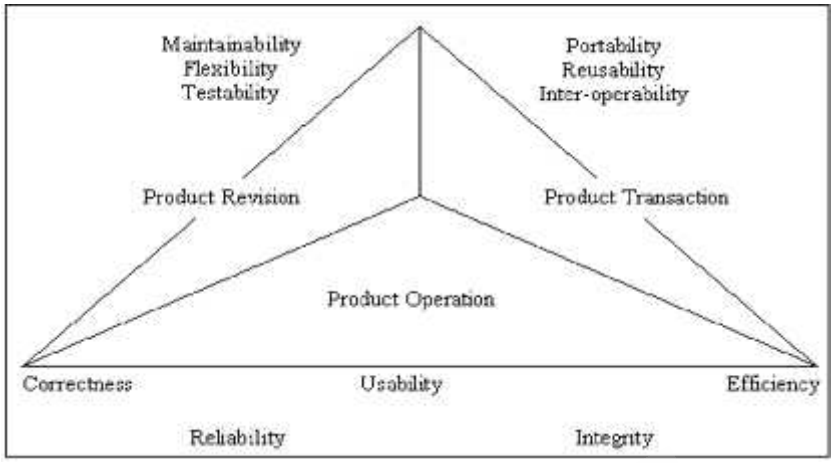
Quality Audits (4.17)

Audits of the quality system must be carried out to ensure that it is effective.

Training (4.18)

Training needs must be identified and met.

Mccall's quality factor:



1.Correctness:

- A software system is expected to meets the explicitly specified functional requirements and the implicitly expected non-functional requirements.
- If a software system satisfies all the functional requirements, the system is said to be correct.

2.Reliability

- Customers may still consider an incorrect system to be reliable if the failure rate is very small and it does not adversely affect their mission objectives.
- Reliability is a customer perception, and an incorrect software can still be considered to be reliable.

3.Efficiency:

- Efficiency concerns to what extent a software system utilizes resources, such as computing power, memory, disk space, communication bandwidth, and energy.
- A software system must utilize as little resources as possible to perform its functionalities.

4.Integrity:

- A system's integrity refers to its ability to withstand attacks to its security.
- In other words, integrity refers to the extent to which access to software or data by unauthorized persons or programs can be controlled.

5.Usability:

- A software is considered to be usable if human users find it easy to use.
- Without a good user interface a software system may fizzle out even if it possesses many desired qualities.

6.Maintainability:

- Maintenance refers to the upkeep of products in response to deterioration of their components due to continuous use of the products.
- Maintenance refers to how easily and inexpensively the maintenance tasks can be performed.
- For software products, there are three categories of maintenance activities : corrective, adaptive and perfective maintenance.

7.Testability:

- Testability means the ability to verify requirements. At every stage of software development, it is necessary to consider the testability aspect of a product.
- To make a product testable, designers may have to instrument a design with functionalities not available to the customer.

8.Flexibility:

- Flexibility is reflected in the cost of modifying an operational system.
- In order to measure the flexibility of a system, one has to find an answer to the question: How easily can one add a new feature to a system.

9.Portability

- Portability of a software system refers to how easily it can be adapted to run in a different execution environment.
- Portability gives customers an option to easily move from one execution environment to another to best utilize emerging technologies in furthering their business.

10.Reusability

- Reusability means if a significant portion of one product can be reused, maybe with minor modifications, in another product.

- Reusability saves the cost and time to develop and test the component being reused.

11.Interoperability :

- Interoperability means whether or not the output of one system is acceptable as input to another system, it is likely that the two systems run on different computers interconnected by a network.
- An example of interoperability is the ability to roam from one cellular phone network in one country to another cellular network in another country.

McCall's Quality Criteria :

A quality criteria is an attribute of a quality factor that is related to software development. For example, modularity is an attribute of the architecture of a software system.

List of McCall's Quality Criteria :

1.Access Audit : Ease with which the software and data can be checked for compliance with standards.

2.Access Control : Provisions for control and protection of the software

3.Accuracy : Precisions of computations and output.

4.Completeness: Degree to which full implementation of required functionalities have been achieved.

5.Communicativeness : Ease with which the inputs and outputs can be assimilated.

6.Conciseness : Compactness of the source code, in terms of lines of code.

7.Consistency : Use of uniform design and implementation techniques.

8.Data commonality : Use of standard data representation.

9.Error tolerance : Degree to which continuity of operation is ensured under adverse conditions.

10.Execution efficiency : Run time efficiency of the software.

11.Expandability : Degree to which storage requirements or software functions can be expanded.

12.Hardware independence : Degree to which a software is dependent on the underlying hardware.

13.Modularity : Provision of highly independent modules.

14.Operability : Ease of operation of the software.

15.Simplicity : Ease with which the software can be understood.

16.Software efficiency : Run time storage requirements of the software.

17.Traceability : Ability to link software components to requirements.

18.Training : Ease with which new users can use the system.

Waterfall model:

The Waterfall Model was 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.

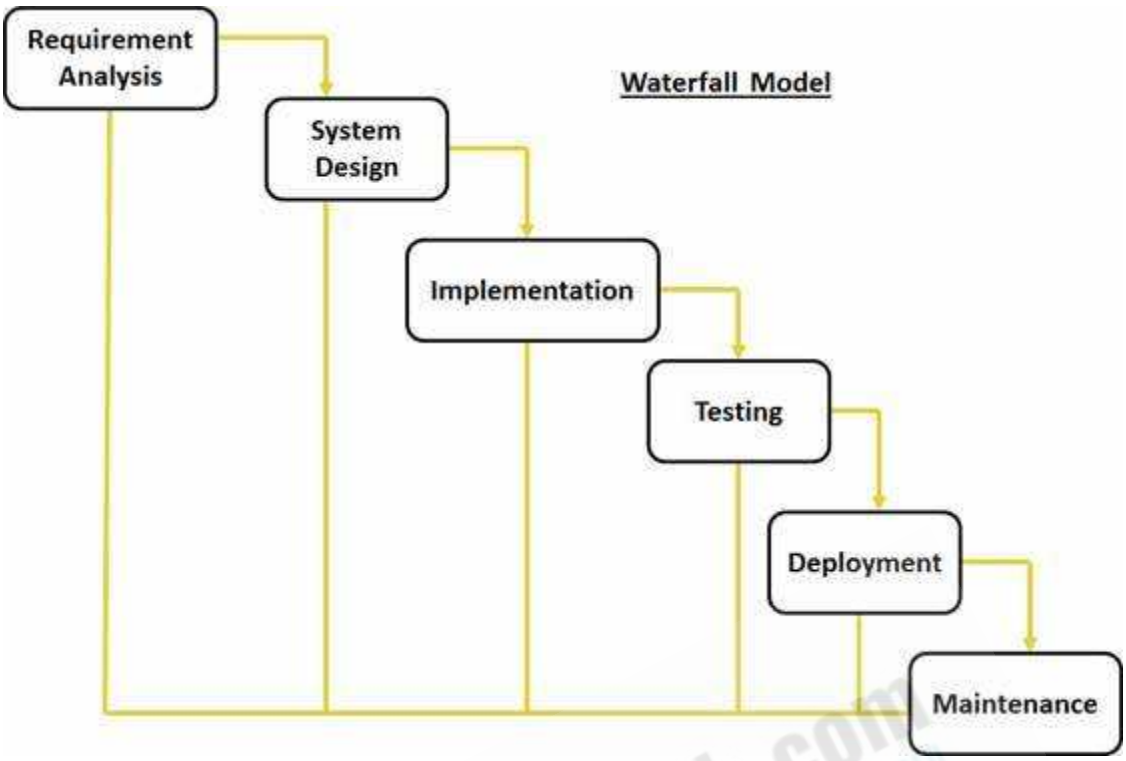
Waterfall model is the earliest SDLC approach that was used for software development.

The waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model. This means that any phase in the development process begins only if the previous phase is complete. In waterfall model phases do not overlap.

Waterfall Model design

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

Following is a diagrammatic representation of different phases of waterfall model.



The sequential phases in Waterfall model are:

- Requirement Gathering and analysis: All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.
- System Design: The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.
- Implementation: With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.
- Integration and Testing: All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

- Deployment of system: Once the functional and non functional testing is done, the product is deployed in the customer environment or released into the market.
- Maintenance: There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model phases do not overlap.

Waterfall Model Application

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are:

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

Waterfall Model Pros & Cons

Advantage

The advantage of waterfall development is that it allows for departmentalization and control. A schedule can be set with deadlines for

each stage of development and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

Disadvantage

The disadvantage of waterfall development is that it does not allow for much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The following table lists out the pros and cons of Waterfall model:

Pros	Cons
• 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.	• 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.

• Well understood milestones. • Easy to arrange tasks. • Process and results are well documented.	• It is difficult to measure progress within stages. • Cannot accommodate changing requirements. • Adjusting scope during the life cycle can end a project. • Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

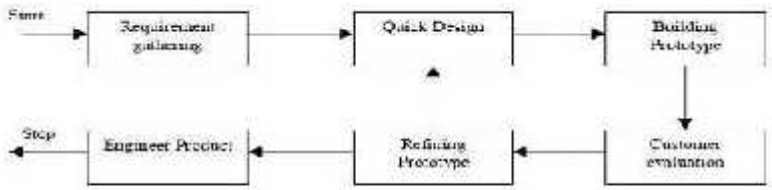
What is Prototype & Spiral model

Protype model:

The basic idea here is that instead of freezing the requirements before a design or coding can proceed, a throwaway prototype is built to understand the requirements. This prototype is developed based on the currently known requirements. By using this prototype, the client can get an "actual feel" of the system, since the interactions with prototype can enable the client to better understand the requirements of the desired system. Prototyping is an attractive idea for complicated and large systems for which there is no manual process or existing system to help determining the requirements.

The prototype are usually not complete systems and many of the details are not built in the prototype. The goal is to provide a system with overall functionality.

Diagram of Prototype model:



Prototyping Model

Advantages of Prototype model:

- Users are actively involved in the development
- Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.
- Errors can be detected much earlier.
- Quicker user feedback is available leading to better solutions.
- Missing functionality can be identified easily
- Confusing or difficult functions can be identified
  Requirements validation, Quick implementation of, incomplete, but functional, application.

Disadvantages of Prototype model:

- Leads to implementing and then repairing way of building systems.
- Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
- Incomplete application may cause application not to be used as the full system was designed
  Incomplete or inadequate problem analysis.

When to use Prototype model:

- Prototype model should be used when the desired system needs to have a lot of interaction with the end users.
- Typically, online systems, web interfaces have a very high amount of interaction with end users, are best suited for Prototype model. It might take a while for a system to be built that allows ease of use and needs minimal training for the end user.
- Prototyping ensures that the end users constantly work with the system and provide a feedback which is incorporated in the prototype to result in a useable system. They are excellent for designing good human computer interface systems.

Spiral Model:

The spiral model is similar to the incremental model, with more emphasis placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model). The baseline spiral, starting in the planning phase, requirements are gathered and risk is assessed. Each subsequent spirals builds on the baseline spiral.

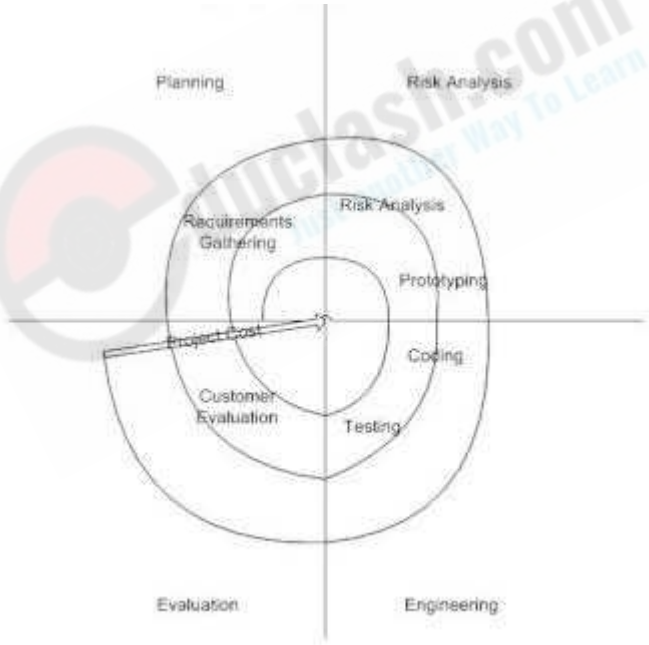
Planning Phase: Requirements are gathered during the planning phase. Requirements like ‘BRS’ that is ‘Bussiness Requirement Specifications’ and ‘SRS’ that is ‘System Requirement specifications’.

Risk Analysis: In the risk analysis phase, a process is undertaken to identify risk and alternate solutions. A prototype is produced at the end of the risk analysis phase. If any risk is found during the risk analysis then alternate solutions are suggested and implemented.

Engineering Phase: In this phase software is developed, along with Testing at the end of the phase. Hence in this phase the development and testing is done.

Evaluation phase: This phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral.

Diagram of Spiral model:



Advantages of Spiral model:

- High amount of risk analysis hence, avoidance of Risk is enhanced.
- Good for large and mission-critical projects.
- Strong approval and documentation control.
- Additional Functionality can be added at a later date.
- Software is produced early in the Software life cycle.

Disadvantages of Spiral model:

- Can be a costly model to use.
- Risk analysis requires highly specific expertise.
- Project's success is highly dependent on the risk analysis phase.
- Doesn't work well for smaller projects.

When to use Spiral model:

- When costs and risk evaluation is important
- For medium to high-risk projects
- Long-term project commitment unwise because of potential changes to economic priorities
- Users are unsure of their needs
- Requirements are complex
- New product line
- Significant changes are expected (research and exploration)

Incremental process model:

The incremental model combines elements of the linear sequential model (applied repetitively) with the iterative philosophy of prototyping. The incremental model applies linear sequences in a staggered fashion as calendar time progresses. Each linear sequence produces a deliverable "increment" of the software
. For example, word-processing software developed using the incremental paradigm might deliver basic file management, editing, and document production functions in the first increment; more sophisticated editing and document production capabilities in the second increment; spelling and grammar checking in the third increment; and advanced page layout capability in the fourth increment. It should be noted that the process flow for any increment can incorporate the prototyping paradigm.
When an incremental model is used, the first increment is often a core product.
That is, basic requirements are addressed, but many supplementary features (some known, others unknown) remain undelivered. The core product is used by the customer (or undergoes detailed review). As a result of use and/or evaluation, a plan is developed for the next increment. The plan addresses the modification of the core product to better meet the needs of the customer and the delivery of additional features and functionality. This process is repeated following the delivery of each increment, until the complete product is produced.

The incremental process model, like prototyping and other evolutionary approaches, is iterative in nature. But unlike prototyping, the incremental model focuses on the delivery of an operational product with each increment. Early increments are stripped down versions of the final product, but they do provide capability that serves the user and also provide a platform for evaluation by the user.
Incremental development is particularly useful when staffing is unavailable for a complete implementation by the business deadline that has been established for the project. Early increments can be implemented with fewer people. If the core product is well received, then additional staff (if required) can be added to implement the next increment. In addition, increments can be planned to manage technical risks.

For example, a major system might require the availability of new hardware that is under development and whose delivery date is uncertain. It might be possible to plan early increments in a way that avoids the use of this hardware, thereby enabling partial functionality to be delivered to end-users without inordinate delay.

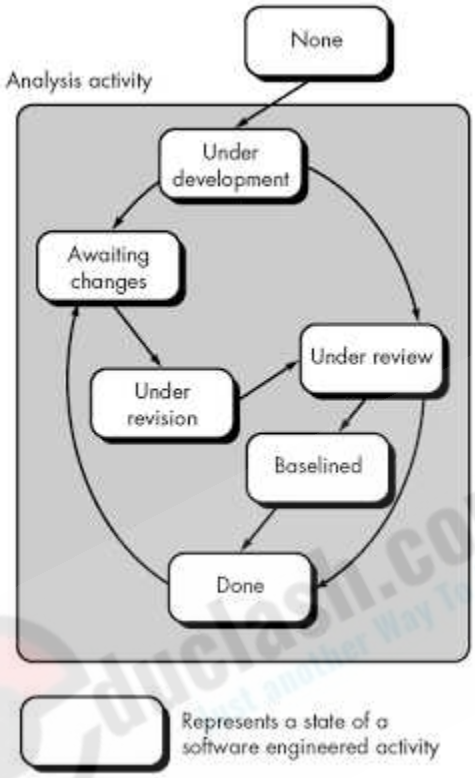
Concurrent Development Model:

The concurrent development model, sometimes called concurrent engineering:

Project managers who track project status in terms of the major phases [of the classic life cycle] have no idea of the status of their projects. These are examples of trying to track extremely complex sets of activities using overly simple models. Note that although . . . [a large] project is in the coding phase, there are personnel on the project involved in activities typically associated with many phases of development simultaneously. For example, personnel are writing requirements, designing, coding, testing, and integration testing [all at the same time]. Software engineering process models by Humphrey and Kellner have shown the concurrency that exists for activities occurring during any one phase. Kellner's more recent work uses statecharts [a notation that represents the states of a process] to represent the concurrent relationship existent among activities associated with a specific event (e.g., a requirements change during late development), but fails to capture the richness of concurrency that exists across all software development and management activities in the project. . . . Most software development process models are driven by time; the later it is, the later in the development process you are. [A concurrent process model] is driven by user needs, management decisions, and review results.

The concurrent process model can be represented schematically as a series of

major technical activities, tasks, and their associated states. For example, the engineering activity defined for the spiral model is accomplished by invoking the following tasks: prototyping and/or analysis modeling, requirements specification, and design.



The activity—analysis—may be in any one of the states noted at any given time. Similarly, other activities (e.g., design or customer communication) can be represented in an analogous manner. All activities exist concurrently but reside in different states. For example, early in a project the customer communication activity (not shown in the figure) has completed its first iteration and exists in the awaiting changes state. The analysis activity (which existed in the none state while initial customer communication was completed) now makes a transition into the under development state. If, however, the customer indicates that changes in requirements must be made, the analysis activity moves from the under development state into the awaiting changes state.

The concurrent process model defines a series of events that will trigger transitions from state to state for each of the software engineering activities. For example, during early stages of design, an inconsistency in the analysis model is uncovered. This generates the event analysis model correction which

will trigger the analysis activity from the done state into the awaiting changes state.

The concurrent process model is often used as the paradigm for the development of client/server applications. A client/server system is composed of a set of functional components. When applied to client/server, the concurrent process model defines activities in two dimensions: a system dimension and a component dimension. System level issues are addressed using three activities: design, assembly, and use. The component dimension is addressed with two activities: design and realization.

Concurrency is achieved in two ways:
(1) system and component activities occur simultaneously and can be modeled using the state-oriented approach described previously;
(2) a typical client/server application is implemented with many components, each of which can be designed and realized concurrently.

In reality, the concurrent process model is applicable to all types of software development and provides an accurate picture of the current state of a project. Rather than confining software engineering activities to a sequence of events, it defines a network of activities. Each activity on the network exists simultaneously with other activities. Events generated within a given activity or at some other place in the activity network trigger transitions among the states of an activity.

Agile Development:

Agile Software development philosophy has it roots in the reality of today's markets. The emergence of agile software processes attempt to deal with the issues introduced by rapidly changing and unpredictable markets. Reading the "Manifesto for Agile Software Development" [3] the basic ideas of the philosophy are introduced through four basic values:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan.

The items on the right have value, however, the items on the left define the agile philosophy. Exploring each of these values will aid in gaining knowledge of the agile process philosophy while exposing how applying the philosophy to defined methods will enhance software development aligning it with today's volatile markets.

Individuals and interactions serve an enhanced role in agile processes. It is a belief among agile process proponents that people can respond quicker and transfer ideas more rapidly when talking face-to-face than they can when reading or writing documentation.

It is an underlying expectation that agile process development occurs in a co-located team environment. An environment where teams work in pairs developing code. The concept of synergy (i.e. The interaction of two or more agents or forces so that their combined effect is greater than the sum of their individual effects.) takes hold because a few designers, sharing a common space, working together, can produce more code quicker than can the same individuals working alone.

Another important aspect of individuals and interactions is that the team is empowered to make all technical decisions. The decision-making process can severely impact a software project because it can be a time consuming process bogged down in a management quagmire. Having the power to make technical decisions in the hands of the technical people is imperative. The implications of this, however, can be far reaching.

Reading [4] we are advised that team members and management must have an equal place in the project. This does not mean that technical people take the role of management. Management still must function as a body to remove roadblocks standing in the way of progress. It is, however, required that management recognize the expertise of the technical team to the point where they are fully empowered to make the technical decisions. Permitting this means, the organization must open other paths of communication between the technical team and the business expertise. Furthermore, the communication must be continuous with no constraints to its access. Decision making, shared responsibility with management and continuous access to business expertise allows individuals to organize around a set of rules operating in an adaptive environment creating innovative and emergent results.

Working Software over comprehensive documentation is a radical concept to many managers in the software industry. This philosophical construct emerges from the idea that the code is the key piece of the documentation. Traditional software processes (big design upfront) on the other hand view the requirements document as the key piece of documentation. A prevailing belief with big design upfront (BDUF) processes is that it is possible to gather all of a customer's requirements, upfront, prior to writing any code.

This is classic engineering. Its success in mechanical and other engineering disciplines makes it attractive to the software industry. Gather all requirements, get a sign-off from the customer and then put into place the procedures (more documentation) to limit and control all changes. The gathering of all requirements up front gives a project a level of predictability and predictability does have value. In fact, software projects that are life

critical must have predictability making the gathering of requirements essential. For many other projects, however, this exercise adds a layer of documentation slowing the project down.

FEASIBILITY STUDY:

A feasibility study is carried out to select the best system that meets performance requirements.

The main aim of the feasibility study activity is to determine whether it would be financially and technically feasible to develop the product. The feasibility study activity involves the analysis of the problem and collection of all relevant information relating to the product such as the different data items which would be input to the system, the processing required to be carried out on these data, the output data required to be produced by the system as well as various constraints on the behaviour of the system.

Technical Feasibility

This is concerned with specifying equipment and software that will successfully satisfy the user requirement. The technical needs of the system may vary considerably, but might include :

- The facility to produce outputs in a given time.
- Response time under certain conditions.
- Ability to process a certain volume of transaction at a particular speed.
- Facility to communicate data to distant locations.

In examining technical feasibility, configuration of the system is given more importance than the actual make of hardware. The configuration should give the complete picture about the system's requirements:

How many workstations are required, how these units are interconnected so that they could operate and communicate smoothly.

What speeds of input and output should be achieved at particular quality of printing.

Economic Feasibility

Economic analysis is the most frequently used technique for evaluating the effectiveness of a proposed system. More commonly known as Cost / Benefit analysis, the procedure is to determine the benefits and savings that are expected from a proposed system and compare them with costs. If benefits outweigh costs, a decision is taken to design and implement the system. Otherwise, further justification or alternative in the proposed system will have to be made if it is to have a chance of being approved. This is an outgoing effort that improves in accuracy at each phase of the system life cycle.

Operational Feasibility

This is mainly related to human organizational and political aspects. The points to be considered are:

• What changes will be brought with the system?

• What organizational structure are disturbed?

• What new skills will be required? Do the existing staff members have these skills? If not, can they be trained in due course of time?

This feasibility study is carried out by a small group of people who are familiar with information system technique and are skilled in system analysis and design process.

Proposed projects are beneficial only if they can be turned into information system that will meet the operating requirements of the organization. This test of feasibility asks if the system will work when it is developed and installed.

What is Software engineering?:

Software engineering is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use.

software engineering has progressed very far in a very short period of time, particularly when compared to classical engineering field (like civil or electrical engineering). In the early days of computing, not much more than 50 years ago, computerized systems were quite small. Most of the programming was done by scientists trying to solve specific, relatively small mathematical problems. Errors in those systems generally had only “annoying” consequences to the mathematician who was trying to find “the answer.” Today we often build monstrous systems, in terms of size and complexity. What is also notable is the progression in the past 50 years of the visibility of the software from mainly scientists and software developers to the general public of all ages. “Today, software is working both explicitly and behind the scenes in virtually all aspects of our lives, including the critical systems that affect our health and well-being.”

Despite our rapid progress, the software industry is considered by many to be in a crisis. Some 40 years ago, the term “Software Crisis” emerged to describe the software industry’s inability to provide customers with high quality products on schedule. “The average software development project overshoots its schedule by half; larger projects generally do worse. And,

some three quarters of all large systems are “operating failures” that either do not function as intended or are not used at all.” (Gibbs, 1994) While the industry can celebrate that software touches nearly all aspects of our daily lives, we can all relate to software availability dates (such as computer games) as moving targets and to computers crashing or locking up. We have many challenges we need to deal with as we continue to progress into a more mature engineering field, one that predictably produces high-quality products.

Evolving role of software:

- Software has become part of the national infrastructure
- Software is the differentiating factor
- Hardware is general-purpose, software is application-specific
- Hardware is becoming a commodity
- Software is often costlier than the hardware it runs on.

Define Computer Software.

A Computer Software is a sequence of instructions written to perform a specified task with a computer.

A computer requires programs to function, typically executing the program’s instructions in a central processor.

The program has an executable form that the computer can use directly to execute the instructions. The same program in its human-readable source code form, from which executable programs are derived enables a programmer to study and develop its algorithms

What are the characteristics of software ? Discuss any three of them

Software is developed, it is not manufactured in the classical sense.

Software doesn’t “wear out”.

Although the industry is moving toward component-based construction, most software continues to be custom built.

SYSTEM SOFTWARE

COLLECTION OF PROFRAMS WRITTEN TO SERVICE OTHER PROGAMS

HEAVY INTERACTION WITH COMPUTER HARDWARE

CONTAINS COMPLEX DATA STRUCTURES AND MULTIPLE EXTERNAL INTERFACES CONCURRENT OPERATION THAT REQUIRES SHEDULING

EXAMPLE:

COMPILERS , EDITORS

FILE MANAGEMENT UTILITIES

OTHER SYSTEM APPLICATIONS

DRIVERS AND NETWORKING SOFTWARE

APPLICATION SOFTWARE

CONSISTS OF STANDALONE PROGRAMS

USED TO SOLVE SPECIFIC BUSINESS NEEDS

PROCESS TECHNICAL DATA/TECHNICAL DECISIONS

CONTROL BUSINESS FUNCTIONS IN REAL TIME

EXAMPLE: CONVENTIONAL DATA PROCESSING APPLICATIONS

REALTIME MANUFACTURING PROCESS CONTROL

PRODUCT LINE SOFTWARE

PROVIDE SPECIFIC CAPABIITY FOR USE BY MANY DIFFERENT CUSTOMERS

FOCUS ON LIMITED & ESOTERIC MARKETPLACE

EXAMPLE:WORD PROCESSING

SPREADSHEETS

COMPUTER GRAPHICS

DATABASE MANAGEMENT

MULTIMEDIA & ENTERTAINMENT

BUSINESS FINANCIAL APPLICATIONS

An Overview of IT Project Management

IT project management is the process of planning, organizing and delineating responsibility for the completion of an organizations' specific information technology (IT) goals.

IT project management includes overseeing projects for software development, hardware installations, network upgrades, cloud computing and virtualization rollouts, business analytics and data management projects and implementing IT services.

In addition to the normal problems that can cause a project to fail, factors that can negatively affect the success of an IT project include advances in technology during the project's execution, infrastructure changes that impact security and data management and unknown dependent relationships among hardware, software, network infrastructure and data. IT projects may also succumb to the first-time, first-use penalty which represents the total risk an organization assumes when implementing a new technology for the first time. Because the technology hasn't been implemented or used before in the organization, there are likely to be complications that will affect the project's likelihood of success.

Managing the Project

These five process groups comprise the project management life cycle and are universal to all projects. The specific phases within a project, however, are unique to each project and represent the project life cycle.

- Initiation – the project goal, need or problem is identified. The project manager is assigned to the project and the project charter is created.
- Planning – the project manager and the project team work together to plan all of the needed steps to reach a successful project conclusion. The project planning processes are iterative in nature and it's expected that planning will happen often throughout the project.
- Execution – once the project plan has been created, the project team goes about executing the project plan to create the deliverables of the project. The project can shift to project planning as needed throughout project execution.

- Monitoring and controlling – as the project is being executed by the project team, the project manager monitors and controls the work for time, cost, scope, quality, risk, and other factors of the project. Monitoring and controlling is also an ongoing process to ensure that the project addresses its targets for each project objective.
- Closing – at the end of each phase and at the end of the entire project, project closure happens to ensure that all of the work has been completed, is approved, and ultimately transferred ownership from the project team to operations.

Managing the Project Knowledge Areas

There are ten project management knowledge areas. These ten knowledge areas segment different actions completed by the project manager throughout the project. The ten project management knowledge areas are:

- Project scope management: the project scope is defined, documented, and approved. The project scope is protected from unauthorized changes, edited with approved changes, and validated by the project stakeholders for project acceptance.
- Project schedule management: the project schedule is defined first by the working hours of the project, any project milestones, and ultimately a project deadline. The project team's availability throughout the project is documented and planned accordingly. The project manager will work with the project team to identify the project tasks and task duration estimates in order to create a project timeline.
- Project costs management: the costs of the project are estimated so that a budget for the project can be assigned. Project costs include materials, services, facilities, software licenses, and other expenses attributed directly to the project.
- Project quality management: what constitutes quality in the project is defined in specific metrics and agreed upon among the stakeholders as early in the project as possible. Quality assurance programs and policies direct the project work, while quality control inspects the project work to confirm that quality has been ascertained in the work.
- Project human resources management: the project manager works with the project team to verify that each team member is completing their assignments, working well with others, and that their participation and performance is reported to their respective managers.
- Project communications management: stakeholders will need information from the project manager will need to provide information to the project manager throughout the project life cycle. This knowledge area create a communications management plan that

address who will need what information, when the information is needed, and the best modality for the communications.

- Project risk management: risks are situations, events, conditions that can threaten, and sometimes benefit, the objectives of the IT project. Risks must be identified, analyzed, and a response created for the risk event. The probability and impact of each risk event is evaluated to create a risk score to justify the costs needed to manage the risk event.
- Project procurement management: should the project need to purchase goods or services, a formal process for procurement will need to be created. The plan should address the project's selection of contract type, administration of the contract, purchasing audits, and contract closeout. Many project managers do not manage procurement, but defer to the organization's centralized procurement or purchasing department and processes.
- Project stakeholder management: stakeholders are anyone that has a vested interest in the project. Stakeholder management is the identification, inclusion, and communication with the groups of project stakeholders. It manages the anxieties and concerns the stakeholders may have about the project work.
- Project integration management: this special knowledge area is the coordination of the events in all of the other knowledge areas. How well the project manager performs in one knowledge directly affects the performance of the other knowledge areas. Project integration management examines the interactions and contingencies among the knowledge areas to ensure that the project is adequately planned, executed, controlled and closed.

These ten knowledge areas are to be managed iteratively throughout the project. With the exception of procurement, a project manager will likely encounter all ten of these knowledge areas in every project. There is no set order in which the areas should be managed, but rather the project manager shifts to the appropriate knowledge and processes based on what's occurring within the project.

IT Project Life Cycle

There are several different approaches to managing an IT project that affect the project life cycle. Organizations can select one of these popular approaches to help reduce the risk of expensive rework, risks from quickly changing technology, or expansive planning at the launch of the project. The project life cycle of a typical IT project moves through iterations of planning, executing, and controlling until the project is ultimately closed and

transferred into operations. However, there are three distinct IT project management life cycles:

Predictive life cycle: this is the most common and traditional project life cycle for IT projects. In this approach the project manager and the project team first define the project scope, project schedule, and expected project costs before the project execution begins. As part of the project planning it's typical for the phases of the project to be defined (each phase does a specific type of project work). In order for the project to move from its initiation to its closure each phase must be started and completed in the specific order as planned. This type of approach is sometimes called a waterfall approach as the project "waterfalls" down the phases of the project.

Iterative life cycle: this approach to IT project management requires that the project management be defined early in the project, but the cost estimates and activity duration estimating are planned at a higher level early in the project. As the project execution occurs costs and duration estimates are created for the most imminent work through iterations of planning. The iterative life cycle also plans for iterations of benefits released to the organization. For example, an iterative life cycle may create a new software with more features with each new release as part of the project.

Adaptive life cycle: this project life cycle also uses an iteration of planning and executing, but the planning that typically last for two weeks. This approach uses a rolling wave of planning and executing through short bursts of both planning and executing. Change is expected in this approach to the IT project and it's ideal for software development project. Agile project management and Scrum are examples of the adaptive life cycle.

All of these life cycles use the concept of phases to move the project work forward. A phase describes the type of work that will take place in that portion of the project. The project manager, the organizational requirements, and even customer requirements can influence what type of project life cycle the project manager will adapt in the project.

JAD (Joint Application Development)

JAD (Joint Application Development) is a methodology that involves the client or end user in the design and development of an application, through a succession of collaborative workshops called JAD sessions. Chuck Morris and Tony Crawford, both of IBM, developed JAD in the late 1970s and began teaching the approach through workshops in 1980.

The JAD approach, in comparison with the more traditional practice, is thought to lead to faster development times and greater client satisfaction, because the client is involved throughout the development process. In comparison, in the traditional approach to systems development, the developer investigates the system requirements and develops an application, with client input consisting of a series of interviews.

A variation on JAD, rapid application development (RAD) creates an application more quickly through such strategies as using fewer formal methodologies and reusing software components.

Types of requirements:

Functional requirements

. Describe what the system should do

A functional requirement document defines the functionality of a system or one of its subsystems. It also depends upon the type of software, expected users and the type of system where the software is used.

Functional user requirements may be high-level statements of what the system should do but functional system requirements should also describe clearly about the system services in detail.

Quality requirements
. Constraints on the design to meet specified levels of quality
Platform requirements
. Constraints on the environment and technology of the system

The project aims at high quality demonstrators. Below we indicate standard for software quality criteria:

Functionality

requirements include suitability, accuracy, interoperability, compliance, security.
- Suitability is evaluated against the validation scenarios that are selected by and used in the project. This requirement is of primary importance.
- Accuracy is evaluated against the intended semantics of the supported administrative procedures. It is imperative that the system functioning respects these procedures. This requirement is of primary importance.
- Interoperability is important where links with existing IT infrastructure and administrations are used. The project only strives for interoperability to the extend that this is needed for the validation scenarios.
- Compliance with standards is only a concern when these standards are operational in certain administrations. The project will itself not enforce any standards.
- Security is a primary concern. User authentication, access limitations and, especially, fire-wall protection barriers for the in-house administration files and databases are required.

Reliability

requirements include maturity, fault tolerance and recoverability.
- Maturity implies that the system is error-proof within the limits of its intended use. Fail-safe capabilities for all uses is not a primary concern unless security is endangered.
- Fault tolerance and recoverability are of primary concern for the interfaces but not for the other parts of the system, unless security is endangered.

Usability

requirements include understandability, learnability and operability.
- Understandability is assured by making the conceptual basis of the system as close as possible to the one familiar to administration officers. This is of primary importance.
- Learnability is important within the scope of the use of the system in the selected scenario validations. Specific measures for learnability are incorporated.
- Language issues are just one aspect of operability. The demonstrators will be developed with interfaces in the language best suited for the end-users concerned. How those can be adapted to deal with multiple languages will be documented and the effort required will be estimated.

Efficiency

requirements can be expressed as time and resource behaviors.
- Controlled time behavior of the client-server part is of primary importance for information access. Time behavior of the communication between the administrations is less crucial. It does not involve a real-time communication line for administration officers, but only for software agents to travel. Time-behavior for external clients is important but not likely to be a problem.
- Excessive resource consumption should not endanger the proper functioning of the normal administrative procedures. Network resource consumption must be kept within limits, which is in itself a motivation for the technology of software agents. The overall external communication cost of running the system is expected to be proportional to the number of cases handled and the number of administrations involved.

Maintainability and Portability

Are important criteria for the project. A solid approach to software development, documentation and explicit recommendations for further development and completion will be an integral part of the output of the project.

Process requirements

. Constraints on the project plan and development methods

Most computer-based systems perform some automated procedures or processes, even the simplest website enables a visitor to navigate around the site to find information. More complex websites authenticate visitors and may automate transactions through e-commerce.

Technical requirements for system procedures and processes identify the non-functional specifications of the proposed system itself. The non-functional requirements can include:

- Performance or speed of the system
- Quality
- Environmental requirements or business rules
- Size
- Ease of use
- Reliability
- Robustness
- Portability

For Example:

- If the system is a sales system: technical requirements may address the number of transactions per minute.
- If the system is a website: the technical requirements may address the page display speed, compatibility with browsers and hardware platforms.
- If the system is a database-centred system: the technical requirements may address the constructs of the database, number of records or processing time.
- If the system is an inventory control system: the technical requirements may address the ability to set the alarm levels for high and low stocks.
- If the system is a network: the technical requirements may address download or response times, application access, redundancy procedures, disk access speeds, number of users etc.

In addition technical requirements for system procedures and processes may address the application architecture, development environment or network topology and protocols.

Software analysis and design includes all activities, which help the transformation of requirement specification into implementation. Requirement specifications specify all functional and non-functional expectations from the software. These requirement specifications come in the shape of human readable and understandable documents, to which a computer has nothing to do.

Software analysis and design is the intermediate stage, which helps human-readable requirements to be transformed into actual code.

Let us see few analysis and design tools used by software designers:

Data Flow Diagram

Data flow diagram is graphical representation of flow of data in an information system. It is capable of depicting incoming data flow, outgoing data flow and stored data. The DFD does not mention anything about how data flows through the system.

There is a prominent difference between DFD and Flowchart. The flowchart depicts flow of control in program modules. DFDs depict flow of data in the system at various levels. DFD does not contain any control or branch elements.

Types of DFD

Data Flow Diagrams are either Logical or Physical.

- Logical DFD - This type of DFD concentrates on the system process, and flow of data in the system.For example in a Banking software system, how data is moved between different entities.
- Physical DFD - This type of DFD shows how the data flow is actually implemented in the system. It is more specific and close to the implementation.

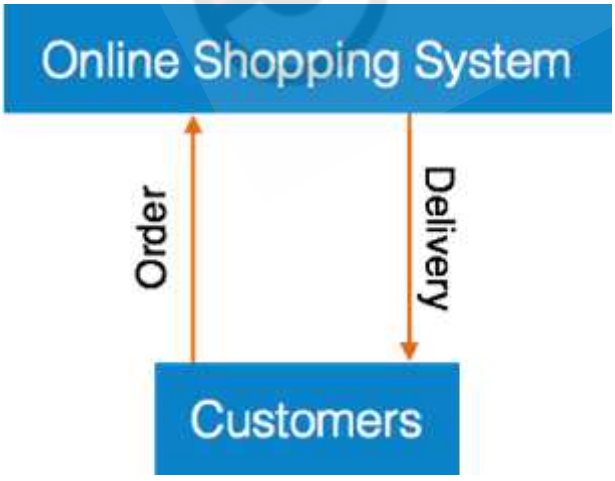
DFD Components

DFD can represent Source, destination, storage and flow of data using the following set of components -

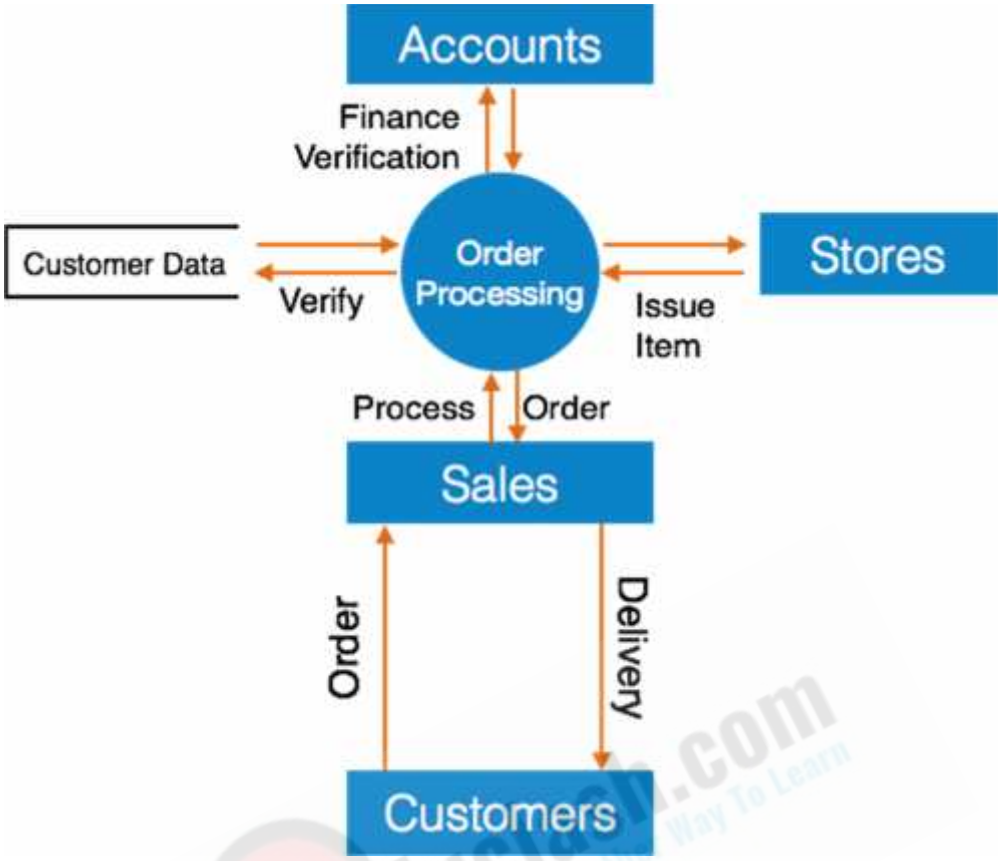
- Entities - Entities are source and destination of information data. Entities are represented by a rectangles with their respective names.
- Process - Activities and action taken on the data are represented by Circle or Round-edged rectangles.
- Data Storage - There are two variants of data storage - it can either be represented as a rectangle with absence of both smaller sides or as an open-sided rectangle with only one side missing.
- Data Flow - Movement of data is shown by pointed arrows. Data movement is shown from the base of arrow as its source towards head of the arrow as destination.

Levels of DFD

- Level 0 - Highest abstraction level DFD is known as Level 0 DFD, which depicts the entire information system as one diagram concealing all the underlying details. Level 0 DFDs are also known as context level DFDs.



- Level 1 - The Level 0 DFD is broken down into more specific, Level 1 DFD. Level 1 DFD depicts basic modules in the system and flow of data among various modules. Level 1 DFD also mentions basic processes and sources of information.



- Level 2 - At this level, DFD shows how data flows inside the modules mentioned in Level 1.

  Higher level DFDs can be transformed into more specific lower level DFDs with deeper level of understanding unless the desired level of specification is achieved.

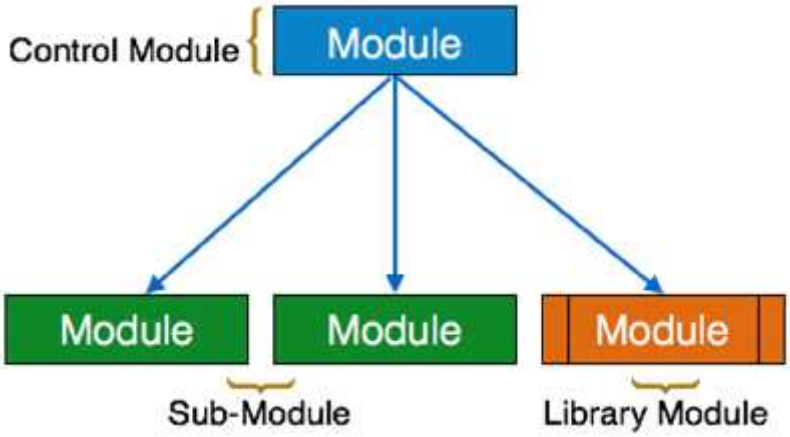
Structure Charts

Structure chart is a chart derived from Data Flow Diagram. It represents the system in more detail than DFD. It breaks down the entire system into lowest functional modules, describes functions and sub-functions of each module of the system to a greater detail than DFD.

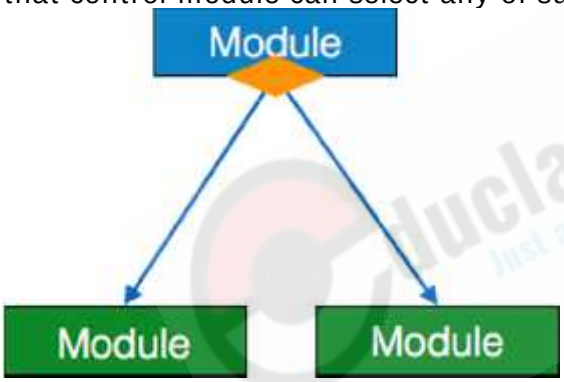
Structure chart represents hierarchical structure of modules. At each layer a specific task is performed.

Here are the symbols used in construction of structure charts -

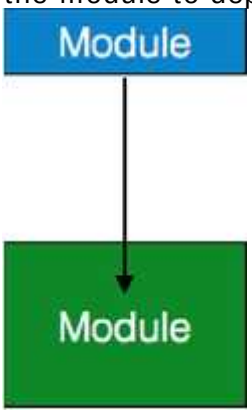
- Module - It represents process or subroutine or task. A control module branches to more than one sub-module. Library Modules are re-usable and invokable from any module.



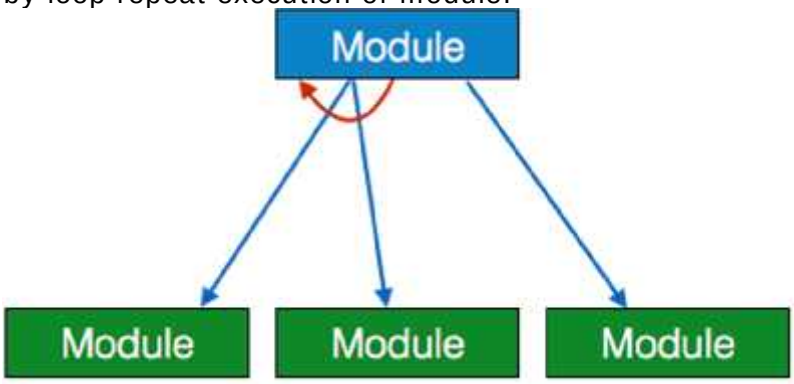
- Condition - It is represented by small diamond at the base of module. It depicts that control module can select any of sub-routine based on some condition.



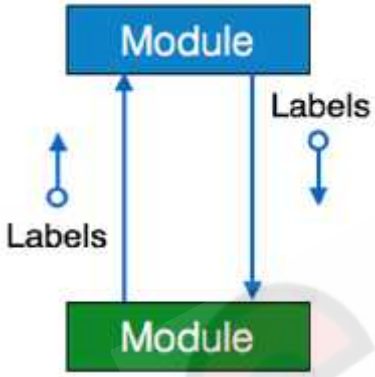
- Jump - An arrow is shown pointing inside the module to depict that the control will jump in the middle of the sub-module.



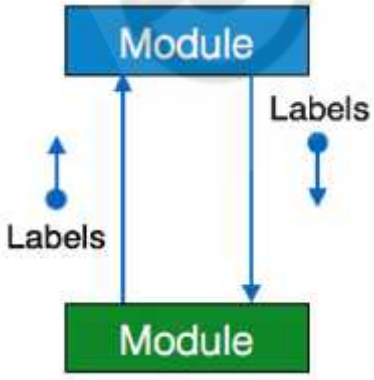
- Loop - A curved arrow represents loop in the module. All sub-modules covered by loop repeat execution of module.



- Data flow - A directed arrow with empty circle at the end represents data flow.



- Control flow - A directed arrow with filled circle at the end represents control flow.



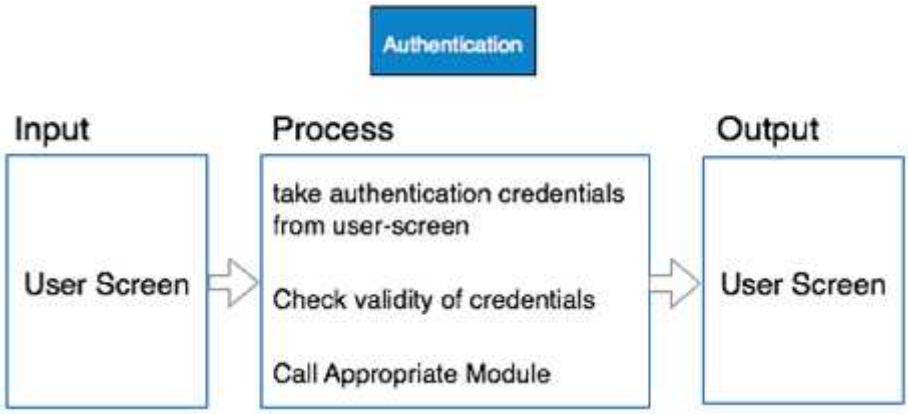
HIPO CHART

HIPO (Hierarchical Input Process Output) diagram is a combination of two organized method to analyze the system and provide the means of documentation. HIPO model was developed by IBM in year 1970.

HIPO diagram represents the hierarchy of modules in the software system. Analyst uses HIPO diagram in order to obtain high-level view of system functions. It decomposes functions into sub-functions in a hierarchical manner. It depicts the functions performed by system.

HIPO diagrams are good for documentation purpose. Their graphical representation makes it easier for designers and managers to get the pictorial idea of the system structure.

In contrast to IPO (Input Process Output) diagram, which depicts the flow of control and data in a module, HIPO does not provide any information about data flow or control flow.



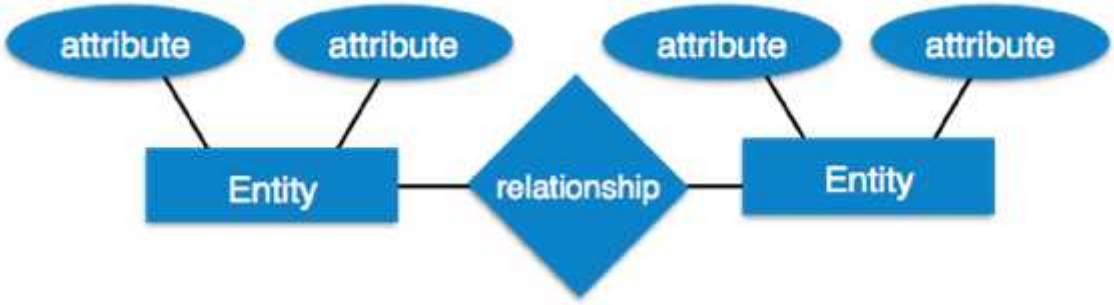
Example

Both parts of HIPO diagram, Hierarchical presentation and IPO Chart are used for structure design of software program as well as documentation of the same.

Entity-Relationship Model

Entity-Relationship model is a type of database model based on the notion of real world entities and relationship among them. We can map real world scenario onto ER database model. ER Model creates a set of entities with their attributes, a set of constraints and relation among them.

ER Model is best used for the conceptual design of database. ER Model can be represented as follows :



- Entity - An entity in ER Model is a real world being, which has some properties called attributes. Every attribute is defined by its corresponding set of values, called domain.

For example, Consider a school database. Here, a student is an entity. Student has various attributes like name, id, age and class etc.

- Relationship - The logical association among entities is called relationship. Relationships are mapped with entities in various ways. Mapping cardinalities define the number of associations between two entities.

  Mapping cardinalities:

  - one to one
  - one to many
  - many to one
  - many to many

Warnier-Orr diagram

Warnier-Orr diagram is a graphic charting technique used in software engineering for system analysis and design purposes. They were initially developed in France by Jean-Dominique Warnier & in the US by Kenneth Orr.This is a kind of hierarchical flowchart that allow the description of the organization of data and procedures.These diagrams are very much similar to the flow charts, but they look like pseudo code rather than blocks and shapes. These are the graphical representations of algorithms. These diagrams help the programmers work on, and record one problem before moving into the next problem or next part. So, that confusion in the complexity of the program will be reduced. This gives the ability to understand the logical structure of the program without getting confused in the complexity of that program. These diagrams are easy to understand and also easy to create. This method aids the design of program structures by identifying the output & processing results & then working backwards to determine the steps & combinations of input needed to produce them. The simple graphic method used in Warnier/Orr diagrams makes the levels in the system evident and the movement of the data between them vivid.

This diagram shows the processes and sequences very well. Each process is very well clearly explained as by having sub processes in it. These are represented by a bracket, grouping its components. Every process is designed in a hierarchical manner.

To construct a WO diagram, one should move backwards and construct it, such as start from the output and should do analysis backwards. There is a advantage of showing data in processes and in steps and sequences and also flow of data from level to level.

Lets have a quick description of elements of WO diagram

Bracket: A bracket encloses a level of decomposition in a diagram. It reveals what something "consists of" at the next level of detail.

Sequence: The sequence of events is defined by the top-to-bottom order in a diagram. That is, an event occurs after everything above it in a diagram, but before anything below it.

OR: You represent choice in a diagram by placing an "OR" operator between the items of a choice. The "OR" operator looks either like or .

AND: You represent concurrency in a diagram by placing an "AND" operator between the concurrent actions. The "AND" operator looks either like or .

Repetition: To show that an action repeats (loops), you simply put the number of repetitions of the action in parentheses below the action.

The diagram below illustrates the use of these constructs to describe a simple process.

You could read the above diagram like this:

"Welcoming a guest to your home (from 1 to many times) consists of greeting the guest and taking the guest's coat at the same time, then showing the guest in. Greeting a guest consists of saying "Good morning" if it's morning, or saying "Good afternoon" if it's

afternoon, or saying "Good evening" if it’s evening. Taking the guest’s coat consists of helping the guest remove their coat, then hanging the coat up."

As you can see, the diagram is much easier to understand than the description.

The Warnier-Orr Diagram is read from left to right and from top to bottom within a bracket.

There are mainly 8 fundamental building blocks in this WO diagram, they are as follows:

· Hierarchy

· Sequence

· Selection

· Complement

· Repetition

· Concurrency

· Begin/End Blocks

· Recursion

Hierarchy:

Hierarchy on a Warnier-Orr Diagram is represented with brackets.

Sequence:

Sequence represents the order in which the tasks to be done, by listing them one below the other.

Selection:

Selection is used to allow choices between more than two tasks.

Complement:

Complement is nothing but the logical not condition.

Repetition:

Repetition provides looping.

Concurrency:

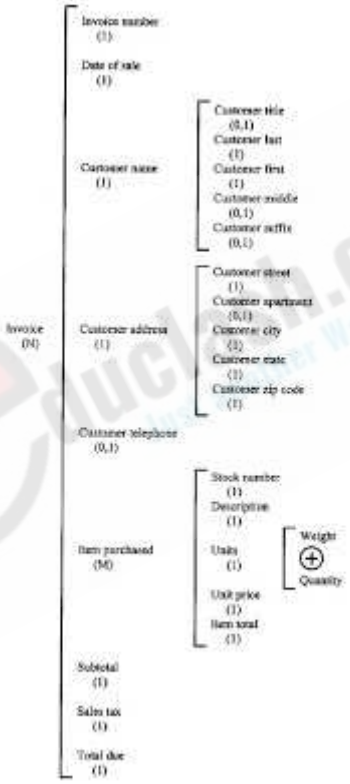
Concurrency allows different tasks to be done at the same time.

Begin/End Blocks:

Here, the Begin block holds the operations performed in the beginning of the task and End block contains operations performed at the end of the task.

Recursive:

Recursive block contains itself a sub process in it.



In this way, any process in any organization can be explained clearly to everyone, because its nothing but the step wise explanation of the task to be performed.

Requirements Elicitation

In requirements engineering, requirements elicitation is the practice of collecting the requirements of a system from users, customers and other stakeholders.[1] The practice is also sometimes referred to as "requirement gathering".

The term elicitation is used in books and research to raise the fact that good requirements cannot just be collected from the customer, as would be indicated by the name requirements gathering. Requirements elicitation is non-trivial because you can never be sure you get all requirements from the user and customer by just asking them what the system should do OR NOT do (for Safety and Reliability). Requirements elicitation practices include interviews, questionnaires, user observation,workshops, brainstorming0, use cases, role playing and prototyping.

Before requirements can be analyzed, modeled, or specified they must be gathered through an elicitation process. Requirements elicitation is a part of the requirements engineering process, usually followed by analysis and specification of the requirements.

Commonly used elicitation processes are the stakeholder meetings or interviews. For example, an important first meeting could be between software engineers and customers where they discuss their perspective of the requirements.

Problems

The requirements elicitation process may appear simple: ask the customer, the users and others what the objectives for the system or product are, what is to be accomplished, how the system or product fits into the needs of business, and finally, how the system or product is to be used on a day-to-day basis. However, issues may arise that complicate the process.

In 1992, Christel and Kang identified problems that indicate the challenges for requirements elicitation:

1. 'Problems of scope'. The boundary of the system is ill-defined or the customers/users specify unnecessary technical detail that may confuse, rather than clarify, overall system objectives.
2. Problems of understanding. The customers/users are not completely sure of what is needed, have a poor understanding of the capabilities and limitations of their computing environment, don't have a full understanding of the problem domain, have trouble communicating needs to the system engineer, omit information that is believed to be "obvious," specify requirements that conflict with the

needs of other customers/users, or specify requirements that are ambiguous or untestable.
3. Problems of volatility. The requirements change over time. The rate of change is sometimes referred to as the level of requirement volatility

Requirements quality can be improved through these approaches:[3]

1. Visualization. Using tools that promote better understanding of the desired end-product such as visualization and simulation.
2. Consistent language. Using simple, consistent definitions for requirements described in natural language and use the business terminology that is prevalent in the enterprise.
3. Guidelines. Following organizational guidelines that describe the collection techniques and the types of requirements to be collected. These guidelines are then used consistently across projects.
4. Consistent use of templates. Producing a consistent set of models and templates to document the requirements.
5. Documenting dependencies. Documenting dependencies and interrelationships among requirements.
6. Analysis of changes. Performing root cause analysis of changes to requirements and making corrective actions.

Guidelines

In 1997, Sommerville and Sawyer suggested a set of guidelines for requirements elicitation, to address concerns such as those identified by Christel and Kang:

- Assess the business and technical feasibility for the proposed system
- Identify the people who will help specify requirements and understand their organizational bias
- Define the technical environment (e.g., computing architecture, operating system, telecommunications needs) into which the system or product will be placed
- Identify "domain constraints" (i.e., characteristics of the business environment specific to the application domain) that limit the functionality or performance of the system or product to be built
- Define one or more requirements elicitation methods (e.g., interviews, focus groups, team meetings)
- Solicit participation from many people so that requirements are defined from different points of view; be sure to identify the rationale for each requirement that is recorded
- Identify ambiguous requirements as candidates for prototyping

- Create usage scenarios or use cases to help customers/users better identify key requirements

Software estimation:

Effective software project estimation is one of the most challenging and important activities in software development. Proper project planning and control is not possible without a sound and reliable estimate. As a whole, the software industry doesn't estimate projects well and doesn't use estimates appropriately. We suffer far more than we should as a result and we need to focus some effort on improving the situation.
Under-estimating a project leads to under-staffing it (resulting in staff burnout), under-scoping the quality assurance effort (running the risk of low quality deliverables), and setting too short a schedule (resulting in loss of credibility as deadlines are missed). For those who figure on avoiding this situation by generously padding the estimate, over-estimating a project can be just about as bad for the organization! If you give a project more resources than it really needs without sufficient scope controls it will use them. The project is then likely to cost more than it should (a negative impact on the bottom line), take longer to deliver than necessary (resulting in lost opportunities), and delay the use of your resources on the next project.

Software Project Estimation 101

The four basic steps in software project estimation are:
1) Estimate the size of the development product. This generally ends up in either Lines of Code
(LOC) or Function Points (FP), but there are other possible units of measure. A discussion of the pros & cons of each is discussed in some of the material referenced at the end of this report.
2) Estimate the effort in person-months or person-hours.
3) Estimate the schedule in calendar months.
4) Estimate the project cost in dollars (or local currency)

Estimating size

An accurate estimate of the size of the software to be built is the first step to an effective estimate. Your source(s) of information regarding the scope of the project should, wherever possible, start with formal descriptions of the requirements - for example, a customer's requirements specification or request for proposal, a system specification, a software requirements specification. If you are [re-]estimating a project in later phases of the project's lifecycle, design documents can be used to provide additional detail. Don't let the lack of a formal scope specification stop you from doing

an initial project estimate. A verbal description or a whiteboard outline are sometimes all you have to start with. In any case, you must communicate the level of risk and uncertainty in an estimate to all concerned and you must re-estimate the project as soon as more scope information is determined.

Two main ways you can estimate product size are:

1) By analogy. Having done a similar project in the past and knowing its size, you estimate each major piece of the new project as a percentage of the size of a similar piece of the previous project.

Estimate the total size of the new project by adding up the estimated sizes of each of the pieces. An experienced estimator can produce reasonably good size estimates by analogy if accurate size values are available for the previous project and if the new project is sufficiently similar to the previous one.

2) By counting product features and using an algorithmic approach such as Function Points to convert the count into an estimate of size. Macro-level "product features" may include the number of subsystems, classes/modules, methods/functions. More detailed "product features" may include the number of screens, dialogs, files, database tables, reports, messages, and so on.

Estimating effort

Once you have an estimate of the size of your product, you can derive the effort estimate. This conversion from software size to total project effort can only be done if you have a defined software development lifecycle and development process that you follow to specify, design, develop, and test the software. A software development project involves far more than simply coding the software – in fact, coding is often the smallest part of the overall effort. Writing and reviewing documentation, implementing prototypes, designing the deliverables, and reviewing and testing the code take up the larger portion of overall project effort. The project effort estimate requires you to identify and estimate, and then sum up all the activities you must perform to build a product of the estimated size.

There are two main ways to derive effort from size:

1) The best way is to use your organization's own historical data to determine how much effort previous projects of the estimated size have taken. This, of course, assumes (a) your organization has been documenting actual results from previous projects, (b) that you have at least one past project of similar size (it is even better if you have several projects of similar size as this reinforces that you consistently need a certain level of effort to develop projects of a given size), and (c) that you will follow a similar development lifecycle, use a similar development methodology, use similar tools, and use a team with similar skills and experience for the new project.

2) If you don't have historical data from your own organization because you haven't started collecting

it yet or because your new project is very different in one or more key aspects, you can use a mature and generally accepted algorithmic approach such as Barry Boehm's COCOMO model or the Putnam Methodology to convert a size estimate into an effort estimate. These models have been derived by studying a significant number of completed projects from various organizations to see how their project sizes mapped into total project effort. These "industry data" models may not be as accurate as your own historical data, but they can give you useful ballpark effort estimates.

Estimating schedule

The third step in estimating a software development project is to determine the project schedule from the effort estimate. This generally involves estimating the number of people who will work on the project, what they will work on (the Work Breakdown Structure), when they will start working on the project and when they will finish (this is the "staffing profile"). Once you have this information, you need to lay it out into a calendar schedule. Again, historical data from your organization's past projects or industry data models can be used to predict the number of people you will need for a project of a given size and how work can be broken down into a schedule.

If you have nothing else, a schedule estimation rule of thumb [McConnell 1996] can be used to get a rough idea of the total calendar time required:

Schedule in months = 3.0 * (effort-months) 1/3

Opinions vary as to whether 2.0 or 2.5 or even 4.0 should be used in place of the 3.0 value – only by trying it out will you see what works for you.

Estimating Cost

There are many factors to consider when estimating the total cost of a project. These include labor, hardware and software purchases or rentals, travel for meeting or testing purposes, telecommunications (e.g., long distance phone calls, video-conferences, dedicated lines for testing, etc.), training courses, office space, and so on.

Exactly how you estimate total project cost will depend on how your organization allocates costs. Some costs may not be allocated to individual projects and may be taken care of by adding an overhead value to labor rates ($ per hour). Often, a software development project manager will only estimate the labor cost and identify any additional project costs not considered "overhead" by the organization.

The simplest labor cost can be obtained by multiplying the project's effort estimate (in hours) by a general labor rate ($ per hour). A more accurate labor cost would result from using a specific labor rate for each staff position (e.g., Technical, QA, Project Management, Documentation, Support, etc.). You would have to determine what percentage of total project effort should be allocated to each position. Again, historical data or industry data models can help.

Working Backwards from Available Time

Projects often have a delivery date specified for them that isn't negotiable - "The new release has to be out in 6 months"; "The customer's new telephone switches go on-line in 12 months and our software has to be ready then". If you already know how much time you have, the only thing you can do is negotiate the set of functionality you can implement in the time available. Since there is always more to do than time available, functionality has to be prioritized and selected so that a cohesive package of software can be delivered on time.

Working backwards doesn't mean you skip any steps in the basic estimation process outlined above. You still need to size the product, although here you really do have to break it down into a number of pieces you can either select or remove from the deliverable, and you still need to estimate effort, schedule, and cost.

This is where estimation tools can be really useful. Trying to fit a set of functionality into a fixed timeframe requires a number of "what if" scenarios to be generated. To do this manually would take too much time and effort. Some tools allow you to play with various options easily and quickly.

Understanding an Estimate's Accuracy

Whenever an estimate is generated, everyone wants to know how close the numbers are to reality. Well, the bottom line is that you won't know exactly until you finish the project – and you will have to live with some uncertainty. Naturally, you will want every estimate to be as accurate as possible given the data you have at the time you generate it. And of course you don't want to present an estimate in a way that inspires a false sense of confidence in the numbers.

What do we mean by an "accurate" estimate? Accuracy is an indication of how close something is to reality. Precision is an indication of how finely something is measured. For example, a size estimate of 70 to 80 KLOC might be both the most accurate and the most precise estimate you can make at the end of the requirements specification phase of a project. If you simplify your size estimate to 75000 LOC it looks more precise, but in reality it's less accurate. If you offer the size estimate as 75281 LOC, it is precise to one LOC but it can only be measured that accurately once the coding phase of the project is completed and an actual LOC count is done.

Business case

A business case captures the reasoning for initiating a project or task. It is often presented in a well-structured written document, but may also sometimes come in the form of a short verbal argument or presentation. The logic of the business case is that, whenever resources such as money or effort are consumed, they should be in support of a specific business need. An example could be that a software upgrade might improve system performance, but the "business case" is that better performance would improve customer satisfaction, require less task processing time, or reduce system maintenance costs. A compelling business case adequately captures both the quantifiable and non-quantifiable characteristics of a proposed project. Business case depends on business attitude and business volume.

Business cases can range from comprehensive and highly structured, as required by formal project management methodologies, to informal and brief. Information included in a formal business case could be the background of the project, the expected business benefits, the options considered (with reasons for rejecting or carrying forward each option), the expected costs of the project, a gap analysis and the expected risks. Consideration should also be given to the option of doing nothing including the costs and risks of inactivity. From this information, the justification for the project is derived. Note that it is not the job of the project manager to build the business case, this task is usually the responsibility of stakeholders and sponsors

Reasons for creating a business case[edit]

Business cases are created to help decision-makers ensure that:

- the proposed initiative will have value and relative priority compared to alternative initiatives based on the objectives and expected benefits laid out in the business case
- the performance indicators found in the business case are identified to be used for proactive realisation of the business and behavioural change.

Development and approval process[edit]

The business case process should be designed to be:

- adaptable - tailored to the size and risk of the proposal
- consistent - the same basic business issues are addressed by every project

- business oriented - concerned with the business capabilities and impact, rather than having a technical focus
- comprehensive - includes all factors relevant to a complete evaluation
- understandable - the contents are clearly relevant, logical and, although demanding, are simple to complete and evaluate
- measurable - all key aspects can be quantified so their achievement can be tracked and measured
- transparent - key elements can be justified directly
- accountable - accountability and commitments for the delivery of benefits and management of costs are clear.

Key Elements of the Business Case Report[edit]

A good business case report, which brings confidence and accountability into the field of making investment decisions, is a compilation of all information collected during enterprise analysis and the business case process. The key purpose is to provide evidence and justification for continuing with the investment proposition. Here is a recommended structure:[2]

- Preface
- Table of Contents
- Executive Briefing
  - Recommendation
  - Summary of Results
  - Decision to be Taken
- Introduction
  - Business Drivers
  - Scope
  - Financial Metrics
- Analysis
  - Assumptions
  - Cash Flow Statement (NPV)
  - Costs
  - Benefits
  - Risk
  - Strategic Options
  - Opportunity Costs
- Conclusion, Recommendation, and Next Steps
- Appendix

Review and approval

At various stages in the project, the business case should be reviewed to ensure that:

- The justification is still valid,
- The project will deliver the solution to the business need.

The result of a review may be the termination or amendment of the project. The business case may also be subject to amendment if the review concludes that the business need has abated or changed, this will have a knock on effect on the project.

Public sector projects

Many public sector projects are now required to justify their need through a business case. In the public sector, the business case is argued in terms of Cost–benefit analysis, which may include both financial and non-financial cost and benefits. This allows the business to take into account societal and environmental benefits, allowing a more comprehensive understanding of economic impacts.

Project selection & approval

Planning process for project selection

- **Strategic planning**, determine the organization's strategy, goals and objectives.
- **Business area analysis**, analyze the business process to achieve strategic goals such improvement of sales, manufacturing, engineering, or IT.
- **Project planning**, define projects that address the strategies.
- **Resource allocation**, choosing which projects to do and assigning resources for working on them.

Project Growth

- billions spent on technology every year
- Sources of project:
  - users
  - top management
  - within information systems
- Process of selecting project
  - idea
  - estimate benefits, costs

Project Motivation

- Cost cutting/avoidance
- Revenue maintenance/enhancement
- Entering new market
- Gaining market share

Measurement of project effect

- INTANGIBLES
  - Difficult to measure cost and benefits
- HIDDEN OUTCOMES
  - time, budget subject to great error
- CHANGE of Information technology
  - technology changes rapidly
    - outdating many good project idea
    - Technology is highly dynamic
- Allowing non-technical staffers to give estimates.
- Underestimating design time and debugging time
- Inadequate/unclear requirements.
- Poorly defined scope of work. This can occur when the work is not broken down far enough or individual elements of work are misinterpreted.
- Omissions. Simply put, you forget something.
- Optimism. This is the rose-colored glasses syndrome, when the all-success scenario is used as the basis for the estimate.
- Failure to assess risk and uncertainty. Neglecting or ignoring risk and uncertainty can result in estimates that are unrealistic.
- Time pressure. If someone comes up to you and says, “Give me a ballpark figure by the end of the day” and “Don’t worry, I won’t hold you to it,” look out! This almost always spells trouble.
- The task performer and the estimator are at two different skill levels
- External pressure.

Failure to involve task performers

Organizational treatment

- Focusing on bottom line justification misleads companies with respect to evaluation of Information technology investment, by treating IS project as capital which requires a rigid cost/benefit analysis.
- The most common justification for project adoption:

- reduction of payroll
- accomplishing a strategic objective (often disregarded because it is difficult to measure)

Initial Risk Value:

Most of these risks have to do with estimating the time, which depends on a number of factors:

- Project manager ability
- experience with application, environment, language
- familiarity with modern programming practice
- availability of critical equipment, software
- completeness of project team
- personnel turnover
- project team size
- relative control of project manager over project team

Selection Methods:

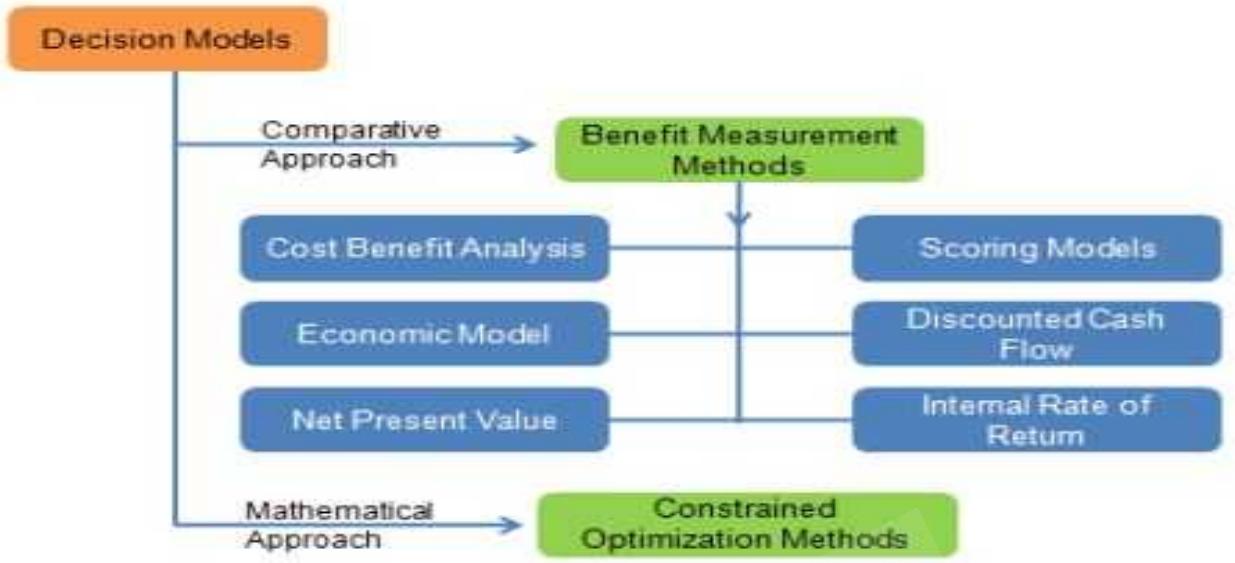
- The comparative approach, the value of one project would need to be compared against the other projects, you could use the benefit measurement methods. This could include various techniques, of which the following are the most common.

- You and your team could come up with certain criteria that you want your ideal project objectives to meet. You could then give each project scores based on how they rate in each of these criteria, and then choose the project with the highest score.

- When it comes to the Discounted Cash flow method, the future value of a project is ascertained by considering the present value and the interest earned on the money. The higher the present value of the project, the better it would be for your organization.

- The rate of return received from the money is what is known as the IRR. Here again, you need to be looking for a high rate of return from the project.

- The mathematical approach is commonly used for larger projects. The constrained optimization methods require several calculations in order to decide on whether or not a project should be rejected.
- - Following is an illustration of two of methods (Benefit Measurement and Constrained Optimization methods).



Project approval Criteria

- Financial
  - net present value/internal rate of return
  - payback
  - profitability index/budgetary constraint
- Management
  - support business objectives
  - respond to competition
  - better decision making
  - satisfy legal requirements
- Development
  - Project needed for effective operation
  - Introduction of new technology

Cost Benefit Analysis

- Cost-benefit analysis (CBA), sometimes called benefit-cost analysis (BCA), is an economic decision-making approach, used particularly in government and business.
- CBA is used in the assessment of whether a proposed project program or policy is worth doing, or to choose between several alternative ones.
- It involves comparing the total expected costs of each option against the total expected benefits, to see whether the benefits outweigh the costs, and by how much.

- In CBA, benefits and costs are expressed in money terms, and are adjusted for the time value of money so that all flows of benefits and flows of project costs over time (which tend to occur at different points in time) are expressed on a common basis in terms of their "present value."

Project Charter

Project Charter refers to a statement of objectives in a project. This statement also sets out detailed project goals, roles and responsibilities, identifies the main stakeholders, and the level of authority of a project manager.

It acts as a guideline for future projects as well as an important material in the organization's knowledge management system.

The project charter is a short document that would consist of new offering request or a request for proposal. This document is a part of the project management process, which is required by Initiative for Policy Dialogue (IPD) and Customer Relationship Management (CRM).

The Role of Project Charter

Following are the roles of a Project Charter:

- It documents the reasons for undertaking the project.
- Outlines the objectives and the constraints faced by the project.
- Provides solutions to the problem in hand.
- Identifies the main stakeholders of the project.

Benefits of Project Charter

Following are the prominent benefits of Project Charter for a project:

- It improves and paves way for good customer relationships.
- Project Charter also works as a tool that improves project management processes.

- Regional and headquarter communications can also be improved to a greater extent.
- By having a project charter, project sponsorship can also be gained.
- Project Charter recognizes senior management roles.
- Allows progression, which is aimed at attaining industry best practices.

Elements in Project Charter

Since project charter is a project planning tool, which is aimed at resolving an issue or an opportunity, the below elements are essential for a good charter project.

For an effective charter project, it needs to address these key elements:

- Identity of the project.
- Time: the start date and the deadline for the project.
- People involved in the project.
- Outlined objectives and set targets.
- The reason for a project charter to be carried out, often referred to as 'business case'.
- Detailed description of a problem or an opportunity.
- The return expected from the project.
- Results that could be expected in terms of performance.
- The expected date that the objectives is to be achieved.
- Clearly defined roles and responsibilities of the participants involved.
- Requirement of resources that will be needed for the objectives to be achieved.
- Barriers and the risks involved with the project.

- Informed and effective communication plan.

Out of all above elements, there are three most important and essential elements that need further elaboration.

Project Scope Management:

In project management, scope refers to all of the products, services, and results provided by your project.

Having a clear and agreed definition of your project's scope and managing that scope as the project proceeds is critical for it's success.

In project management, there are actually two types of scope you need to be familiar with...

- Product Scope
- Project Scope

Product scope is what your end customer cares about. It is all of the features and functions that the product or service your project is delivering will provide.

Project scope is all of the work that needs to be done to provide the product or service your project is delivering.

Now that you know what project management scope is, there are three areas that you will need to focus on to make sure you are managing your project's scope effectively...

- Scope Planning
- Scope Control
- Scope Verification

Scope Planning

The purpose of scope planning is to clearly define what is included in the project. Just as importantly, in the process of defining what's included you will also explicitly state what is excluded from the project.

Scope planning has three steps...

Step 1: Collect Requirements

The first step in scope planning is to determine the needs and expectations of the key project stakeholders in order to meet the project objectives. These are the requirements that your project will need to fulfill in order for your stakeholders to consider your project a success.

Some approaches you can use to collect requirements are...

- Interviews
- Focus Groups
- Workshops

Using these approaches, you will be able to capture the requirements from the different project stakeholders that the project will need to meet.

Step 2: Define Scope

Once you've collected the requirements, you can use them to develop your Project Scope Statement. The Project Scope Statement provides a detailed description of everything included in the project.

Some of the critical items addressed in the Project Scope Statement are...

- Product Scope Description
- Product Acceptance Criteria
- Project Deliverables
- Project Exclusions
- Project Constraints
- Project Assumptions

Once agreed with the key stakeholders, the Project Scope Statement becomes a basis for project decisions.

Step 3: Create the Work Breakdown Structure

The final step in Scope Planning is to create the work breakdown structure (WBS) for the project. A WBS is used to breakdown each project deliverable identified in the Project Scope Statement into smaller, more manageable components.

For example, if you are responsible for setting up a conference for a new product your company is launching, one of the deliverables you would need

to handle would be "Venue." Venue as a deliverable is quite large, so it should be broken down...

- Book Venue
  - Identify potential venues
  - Decide venue
- Power
- Sound & Lighting
- Tables & Chairs
- Catering

If needed, each component can be broken down further just as how "Book Venue" was.

A work breakdown structure will help you estimate your project's cost, understand what skills you will need your team to have, and determine how many people you will need to deliver the project. The WBS will also help when you start building your schedule.

Scope Control

Scope control is the process of managing the scope of your project so that any changes to the project scope are handled in a controlled way. This implies that you will need to set up a change control process.

It would be great if once the scope was agreed during the scope planning, that it wouldn't change. But in the real world, unforeseen events can make a change in scope necessary. As long as changes are handled in a controlled way, it isn't a bad thing. When changes are uncontrolled, it is known as scope creep.

A controlled change would come via a formal change request. The request would be analyzed to determine the impact to the projects schedule, cost, and quality. The stakeholders involved in the change control process can then decide whether they can accept the impacts to the original plan. If it is accepted, then the project manager adjusts the plans and baselines accordingly.

Scope Verification

Scope verification is where the project deliverables are reviewed by the customer to make sure they have been completed according to the acceptance criteria defined during scope planning.

Your ability to effectively perform project management scope planning, control, and verification is important for ensuring your project is successful.

Relationship between people and Effort

Software engineering handles the relationship between people and effort management for product development phase. These some points are as follows:-

1. When software size is small single person can handle same project by performing steps like requirement analysis, designing, code generation, and testing etc.
2. If the project is large additional people are required to complete. the project in stipulated in time it become easy to complete project by distributing work among people and get it done as early as possible.
3. The communication path of new comer also increase as time increase and day by day the project become extra complicated. And new customer gets confusion become more after the days by days.
4. It is possible to reduce a desire project completion date by getting more people to same point. It also possible to expand the completion date by reducing number of resources. And you can mention for the date of completion.
5. The Putnam Norden Rayleigh (PNR) curve is an indication of relationship which exists between effort applied and delivery time for software project.
6. The curve indicate a minimum time value at to which indicates test cost time for delivery as use move to left to right. It is observed that curved raised non-linearly.
7. It is possible to make delivery fast; the curve rises very sharply to left of td. The pnr curve indicates that project delivery time should not be compressed much behind on td.
8. The number of delivery lines of code are also known as source statements L. Relationship of L with effort & development time by equation can be described as

L= P * E1/3 T ¾

Here ‘E’ represents development effort in person months, P is productivity

1. After rearranging the last equation can arrive at an expansion for development effort e.

E = L3/(P3 T4)

E is called as effort expanded over entire life cycle for software development and maintenance

T is the development period in years. And this equation is lead to

E = L3 / (P3 T4) ~ 3.8 Person years.

This shows that by extending last date of project with six month e.g. we can reduce the no of people from eight to four. The outcome benefit can be gained by using less number of people over longer time to achieve the same objective.

These are all about the relationship between people and effort in software engineering.

Relationship between people and effort

Description of People and process management spectrum

The people:-

People management capability maturity model (PM-CMM) has been developed by software engineering institute.

The people management capability maturity model defines following key practice area for software people:-

- Recruiting
- Selection
- Performance
- Training
- Team / culture development
- Carrier development

It guides organizations in a creation of mature software process. The various groups are involved for the most needed communication and co-ordination issue required for any effective software. These groups can be categorized as under:-

1) Stake holder: – Those people who are involved in software process and every software project. Stake holder can be senior manager, project manager practitioner's customer and end user.

2) Team leader: – The MOI model of leadership state characteristics that define and effective project manager motivation, organization and ideas. Successful project leaders use problem solving strategy.

3) Software team:- The people directly involved in a software project are within the software project manager scope seven project factor should be considered when planning structure of software engineering team these are as follows:-

- Difficult of problem to be solved.
- Size of resultant program.
- Time that team will stay together.
- The rigidity of a delivery date.
- The degree of communication required for project.

4) Agile teams: it is very active team which is a very small highly motivated project team consist of a informal method which result into overall development simplicity. These are all about the people description.

The Process:-

A comprehensive plan for a software development can be established from a software process framework. A number of a different task based milestone, work product and quality assurance points enables the frameworks activity to be adapted to characteristics of a software project. The frame work activity that characteristics the software processes are applicable to all software projects. The problem is to be selected process model that is appropriate for software to be a engineered by a software team.

The project manager must decide which process mode is appropriate for:

1) Customer who has a request product

2) the characteristics of product

3) the project environment.

Once the primary plan is established process decomposition begins a complete plan reflecting work task required to populate to require to framework activities must be created. These are all about process.

Process Decomposition:-

A software team should have extent of adaptability in choosing the software process model which is a best for a project and engineering task. A relatively small project might be a best accomplished using linear frequent approach. If very tight time constraints are imposed then problem can be a heavily compartmentalization RAD model. Project with other characteristics will lead to the selection of a other process model. Once a process model and process framework is decided then generic communication framework communication, planning, modeling, construction and deployment can be used. It will works for a linear model for iterative and incremental model, for evolutionary model and for concurrent or concurrent assembly models.

Process decomposition commence when the project manager asks how we accomplish the actual activity. Process decomposition is appropriate for the small relatively and a simple project for it. It should be noted work task s must be adapted to specific need of a project. These are about the process decomposition in the software engineering for product development activities.

Critical Path Method:

If you have been into project management, I'm sure you have already heard the term 'critical path method.'

If you are new to the subject, it is best to start with understanding the 'critical path' and then move on to the 'critical path method.'

Critical path is the sequential activities from start to the end of a project. Although many projects have only one critical path, some projects may have more than one critical paths depending on the flow logic used in the project.

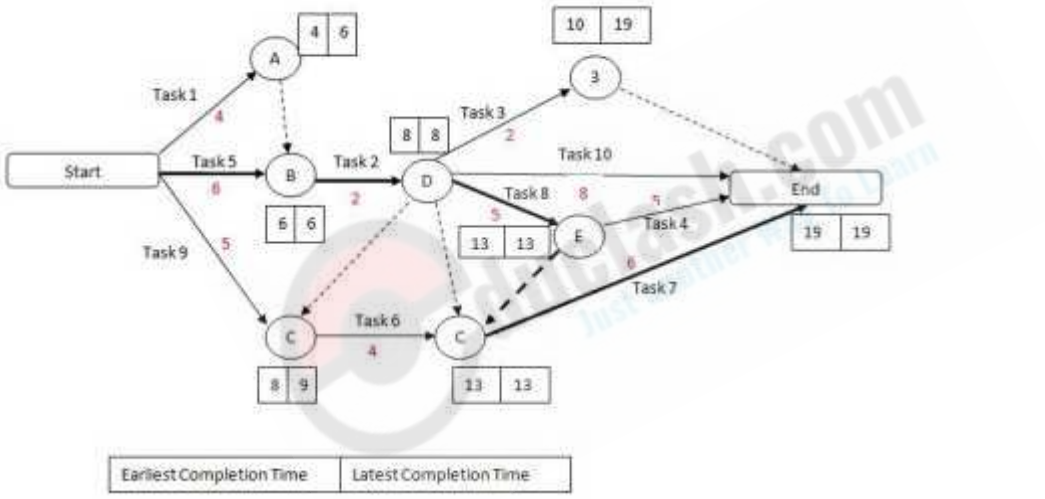
If there is a delay in any of the activities under the critical path, there will be a delay of the project deliverables.

Most of the times, if such delay is occurred, project acceleration or re-sequencing is done in order to achieve the deadlines.

Critical path method is based on mathematical calculations and it is used for scheduling project activities. This method was first introduced in 1950s as a joint venture between Remington Rand Corporation and DuPont Corporation.

The initial critical path method was used for managing plant maintenance projects. Although the original method was developed for construction work, this method can be used for any project where there are interdependent activities.

In the critical path method, the critical activities of a program or a project are identified. These are the activities that have a direct impact on the completion date of the project.



Key Steps in Critical Path Method

Let's have a look at how critical path method is used in practice. The process of using critical path method in project planning phase has six steps.

Step 1: Activity specification

You can use the Work Breakdown Structure (WBS) to identify the activities involved in the project. This is the main input for the critical path method.

In activity specification, only the higher-level activities are selected for critical path method.

When detailed activities are used, the critical path method may become too complex to manage and maintain.

Step 2: Activity sequence establishment

In this step, the correct activity sequence is established. For that, you need to ask three questions for each task of your list.

- Which tasks should take place before this task happens.
- Which tasks should be completed at the same time as this task.
- Which tasks should happen immediately after this task.

Step 3: Network diagram

Once the activity sequence is correctly identified, the network diagram can be drawn (refer to the sample diagram above).

Although the early diagrams were drawn on paper, there are a number of computer softwares, such as Primavera, for this purpose nowadays.

Step 4: Estimates for each activity

This could be a direct input from the WBS based estimation sheet. Most of the companies use 3-point estimation method or COCOMO based (function points based) estimation methods for tasks estimation.

You can use such estimation information for this step of the process.

Step 5: Identification of the critical path

For this, you need to determine four parameters of each activity of the network.

- Earliest start time (ES) - The earliest time an activity can start once the previous dependent activities are over.
- Earliest finish time (EF) - ES + activity duration.
- Latest finish time (LF) - The latest time an activity can finish without delaying the project.

- Latest start time (LS) - LF - activity duration.

The float time for an activity is the time between the earliest (ES) and the latest (LS) start time or between the earliest (EF) and latest (LF) finish times.

During the float time, an activity can be delayed without delaying the project finish date.

The critical path is the longest path of the network diagram. The activities in the critical path have an effect on the deadline of the project. If an activity of this path is delayed, the project will be delayed.

In case if the project management needs to accelerate the project, the times for critical path activities should be reduced.

Step 6: Critical path diagram to show project progresses

Critical path diagram is a live artefact. Therefore, this diagram should be updated with actual values once the task is completed.

This gives more realistic figure for the deadline and the project management can know whether they are on track regarding the deliverables.

Advantages of Critical Path Method

Following are advantages of critical path methods:

- Offers a visual representation of the project activities.
- Presents the time to complete the tasks and the overall project.
- Tracking of critical activities.

Basic Planning Purchases and Acquisitions

The Plan Purchases and Acquisitions process identifies which project needs can best be met by purchasing or acquiring products, services, or results outside the project organization, and which project needs can be accomplished by the project team during project execution. This process involves consideration of whether, how, what, how much, and when to acquire.

When the project obtains products, services, and results required for project performance from outside the performing organization, the processes from Plan Purchases and Acquisitions through Contract Closure are performed for each item to be purchased or acquired.

The Plan Purchases and Acquisitions process also includes consideration of potential sellers, particularly if the buyer wishes to exercise some degree of influence or control over contracting decisions. Consideration should also be given to who is responsible for obtaining or holding any relevant permits and professional licenses that may be required by legislation, regulation, or organizational policy in executing the project.

The project schedule can significantly influence the Plan Purchases and Acquisitions process. Decisions made in developing the procurement management plan can also influence the project schedule and are integrated with Schedule Development Activity Resource Estimating, and make- or-buy decisions.

The Plan Purchases and Acquisitions process includes reviewing the risks involved in each make-or-buy decision; it also includes reviewing the type of contract planned to be used with respect to mitigating risks and transferring risks to the seller.

Plan Purchases and Acquisitions: Inputs

.1 Enterprise Environmental Factors

Enterprise environmental factors that are considered include the conditions of the marketplace and what products, services, and results are available in the marketplace, from whom and under what terms and conditions. If the performing organization does not have formal purchasing or contracting groups, then the project team will have to supply both the resources and the expertise to perform project procurement activities.

.2 Organizational Process Assets

Organizational process assets provide the existing formal and informal procurement-related policies, procedures, guidelines, and management systems that are considered in developing the procurement management plan and selecting the contract types to be used. Organizational policies frequently constrain procurement decisions. These policy constraints can include limiting the use of simple purchase orders and requiring all purchases above a certain value to use a longer form of contract, requiring specific forms of contracts, limiting the ability to make specific make-or-buy decisions, and limiting, or requiring, specific types or sizes of sellers.

Organizations in some application areas also have an established multi-tier supplier system of selected and pre-qualified sellers to reduce the number of direct sellers to the organization and establish an extended supply chain.

.3 Project Scope Statement

The project scope statement describes the project boundaries, requirements, constraints, and assumptions related to the project scope. Constraints are specific factors that can limit both the buyer's and seller's options. One of the most common constraints for many projects is availability of funds. Other constraints can involve required delivery dates, available skilled resources, and organizational policies. Assumptions are factors that will be considered to be true, and which can include items such as the assumed availability of multiple sellers or a sole-source seller. Requirements with contractual and legal implications can include health, safety, security, performance, environmental, insurance, intellectual property rights, equal employment opportunity, licenses, and permits.

The project scope statement provides important information about project needs and strategies that are considered during the Plan Purchases and Acquisitions process. The project scope statement also provides the list of deliverables and acceptance criteria for the project and its products, services, and results. Consideration is given to all such factors that may need to be included in the procurement documentation and flowed down within a contract to sellers.

The product scope description component of the project scope statement provides important information about any technical issues or concerns related to the products, services, and results of the project that are considered during the Plan Purchases and Acquisitions process.

The work breakdown structure (WBS) and WBS dictionary components of the project scope statement provide the structured and detailed plan for the project scope:

.4 Work Breakdown Structure

The Work Breakdown Structure provides the relationship among all the components of the project and the project deliverables.

.5 WBS Dictionary

The WBS dictionary (Section 5.3.3.3) provides detailed statements of work that provide an identification of the deliverables and a description of the work within each WBS component required to produce each deliverable.

.6 Project Management Plan

The project management plan provides the overall plan for managing the project and includes subsidiary plans such as a scope management plan, procurement management plan, quality management plan, and contract management plans, which provide guidance and direction for procurement management planning. To the extent that other planning outputs are available, those other planning outputs are considered during the Plan Purchases and Acquisitions process. Other planning outputs that are often considered include:

- Risk register . Contains risk-related information such as the identified risks, risk owners, and risk responses.
- Risk-related contractual agreements Includes agreements for insurance, services, and other items as appropriate, that are prepared to specify each party's responsibility for specific risks, should they occur.
- Activity resource requirements.
- Project schedule.
- Activity cost estimates
- Cost baseline

Tools and Techniques

.1 Make-or-Buy Analysis

The make-or-buy analysis is a general management technique and a part of the project Plan Purchases and Acquisition process that can be used to determine whether a particular product or service can be produced by the project team or can be purchased. Any project budget constraints are factored in the make-or-buy decisions. If a buy decision is to be made, then a further decision of whether to purchase or rent is also made. The analysis includes both indirect as well as direct costs. For example, the buy-side of

the analysis includes both the actual out-of- pocket costs to purchase the product as well as the indirect costs of managing the purchasing process.

In a make-or-buy analysis, if a buy decision is to be made, it also reflects the perspective of the project team's organization as well as the immediate needs of the project. For example, purchasing an item (anything from a construction crane to a personal computer) rather than renting or leasing it may or may not be cost effective from the perspective of the project. However, if the project team's organization has an ongoing need for the item, the portion of the purchase cost allocated to the project could be less than the cost of the rental. The cost allocation could be based upon a margin analysis.

The long-range strategy of the project team's organization is also a component in the make-or-buy analysis. Items needed for the performance of the project may not be available within the organization. However, the organization may anticipate future requirements for those items and the organization's plans may also be based on making the items in the future. Such considerations can lead to a make decision in spite of the current project constraints and requirements. When this occurs, the costs charged to the project can be less than the actual costs, with the difference representing the organization's investment for the future.

.2 Expert Judgment

Expert technical judgment will often be required to assess the inputs to and outputs from this process. Expert purchasing judgment can also be used to develop or modify the criteria that will be used to evaluate offers or proposals made by sellers. Expert legal judgment may involve the services of a lawyer to assist with non- standard procurement terms and conditions. Such judgment and expertise, including business expertise and technical expertise, can be applied to both the technical details of the procured products, services, or results and to various aspects of the procurement management processes.

.3 Contract Types

Different types of contracts are more or less appropriate for different types of purchases. The type of contract used and the specific contract terms and conditions set the degree of risk being assumed by both the buyer and seller. Contracts generally fall into one of three broad categories:

- Fixed-price or lump-sum contracts. This category of contract involves a fixed total price for a well-defined product. Fixed-price

contracts can also include incentives for meeting or exceeding selected project objectives, such as schedule targets. The simplest form of a fixed-price contract is a purchase order for a specified item to be delivered by a specified date for a specified price.

- Cost-reimbursable contracts. This category of contract involves payment (reimbursement) to the seller for seller's actual costs, plus a fee typically representing seller profit. Costs are usually classified as direct costs or indirect costs. Direct costs are costs incurred for the exclusive benefit of the project (e.g., salaries of full-time project staff). Indirect costs, also called overhead and general and administrative costs, are costs allocated to the project by the project team as a cost of doing business (e.g., salaries of management indirectly involved in the project, cost of electric utilities for the office). Indirect costs are usually calculated as a percentage of direct costs. Cost-reimbursable contracts often include incentive clauses where if the seller meets or exceeds selected project objectives, such as schedule targets or total cost, then the seller receives an incentive or bonus payment. Three common types of cost-reimbursable contracts are CPF, CPFF, and CPIF.
    1. Cost-Plus-Fee (CPF) or Cost-Plus-Percentage of Cost (CPPC). Seller is reimbursed for allowable costs for performing the contract work and receives a fee calculated as an agreed-upon percentage of the costs. The fee varies with the actual cost.
    2. Cost-Plus-Fixed-Fee (CPFF). Seller is reimbursed for allowable costs for performing the contract work and receives a fixed fee payment calculated as a percentage of the estimated project costs. The fixed fee does not vary with actual costs unless the project scope changes.
    3. Cost-Plus-Incentive-Fee (CPIF). Seller is reimbursed for allowable costs for performing the contract work and receives a predetermined fee, an incentive bonus, based upon achieving certain performance objective levels set in the contract. In some CPIF contracts, if the final costs are less than the expected costs, then both the buyer and seller benefit from the cost savings based upon a pre-negotiated sharing formula.
- Time and Material (T&M) contracts. T&M contracts are a hybrid type of contractual arrangement that contains aspects of both cost-reimbursable and fixed-price type arrangements. These types of contracts resemble cost- reimbursable type arrangements in that they are open ended. The full value of the agreement and the exact quantity of items to be delivered are not defined by the buyer at the time of the contract award. Thus, T&M contracts can grow in contract value as if they were cost-reimbursable type arrangements.

Conversely, T&M arrangements can also resemble fixed-price arrangements. For example, unit rates can be preset by the buyer and seller when both parties agree on the rates for a specific resource category.

The requirements (e.g., standard or custom product version, performance reporting, cost data submittals) that a buyer imposes on a seller, along with other planning considerations such as the degree of market competition and degree of risk, will also determine which type of contract will be used. In addition, the seller can consider some of those specific requirements as items that have additional costs. Another consideration relates to the future potential purchase of the product or service being acquired by the project team. Where such potential can be significant, sellers may be inclined or induced to charge prices that are less than would be the case without such future sale potential. While this can reduce the costs to the project, there are legal ramifications if the buyer promises such potential and it is not, in fact, realized.

Plan Purchases and Acquisitions: Outputs

.1 Procurement Management Plan

The procurement management plan describes how the procurement processes will be managed from developing procurement documentation through contract closure. The procurement management plan can include:

- Types of contracts to be used
- Who will prepare independent estimates and if they are needed as evaluation criteria
- Those actions the project management team can take on its own, if the performing organization has a procurement, contracting, or purchasing department
- Standardized procurement documents, if they are needed
- Managing multiple providers
- Coordinating procurement with other project aspects, such as scheduling and performance reporting
- Constraints and assumptions that could affect planned purchases and acquisitions
- Handling the lead times required to purchase or acquire items from sellers and coordinating them with the project schedule development
- Handling the make-or-buy decisions and linking them into the Activity Resource Estimating and Schedule Development processes

- Setting the scheduled dates in each contract for the contract deliverables and coordinating with the schedule development and control processes
- Identifying performance bonds or insurance contracts to mitigate some forms of project risk
- Establishing the direction to be provided to the sellers on developing and maintaining a contract work breakdown structure
- Establishing the form and format to be used for the contract statement of work
- Identifying pre-qualified selected sellers, if any, to be used
- Procurement metrics to be used to manage contracts and evaluate sellers.

A procurement management plan can be formal or informal, can be highly detailed or broadly framed, and is based upon the needs of the project. The procurement management plan is a subsidiary component of the project management plan.

.2 Contract Statement of Work

Each contract statement of work defines, for those items being purchased or acquired, just the portion of the project scope that is included within the related contract. The statement of work (SOW) for each contract is developed from the project scope statement, the project work breakdown structure (WBS), and WBS dictionary. The contract SOW describes the procurement item in sufficient detail to allow prospective sellers to determine if they are capable of providing the item. Sufficient detail can vary, based on the nature of the item, the needs of the buyer, or the expected contract form. A contract SOW describes the products, services, or results to be supplied by the seller. Information included in a contract SOW can include specifications, quantity desired, quality levels, performance data, period of performance, work location, and other requirements.

The contract SOW is written to be clear, complete, and concise. It includes a description of any collateral services required, such as performance reporting or post-project operational support for the procured item. In some application areas, there are specific content and format requirements for a contract SOW. Each individual procurement item requires a contract SOW. However, multiple products or services can be grouped as one procurement item within a single contract SOW.

The contract SOW can be revised and refined as required as it moves through the procurement process until incorporated into a signed contract.

For example, a prospective seller can suggest a more efficient approach or a less costly product than that originally specified.

.3 Make-or-Buy Decisions

The documented decisions of what project products, services, or results will be either be acquired or will be developed by the project team. This may include decisions to buy insurance policies or performance bonds contracts to address some of the identified risks. The make-or-buy decisions document can be as simple as a listing that includes a short justification for the decision. These decisions can be iterative as subsequent procurement activities indicate a need for a different approach.

.4 Requested Changes

Requested changes to the project management plan and its subsidiary plans and other components may result from the Plan Purchases and Acquisition process. Requested changes are processed for review and disposition through the Integrated Change Control process.

SEI = 'Software Engineering Institute' at Carnegie-Mellon University; initiated by the U.S. Defense Department to help improve software development processes.

McCalls Quality Model

Jim McCall produced this model for the US Air Force and the intention was to bridge the gap between users and developers. He tried to map the user view with the developer's priority.

McCall identified three main perspectives for characterizing the quality attributes of a software product.

These perspectives are:-

- Product revision (ability to change).
- Product transition (adaptability to new environments).
- Product operations (basic operational characteristics).

Product revision

The product revision perspective identifies quality factors that influence the ability to change the software product, these factors are:-

- Maintainability, the ability to find and fix a defect.
- Flexibility, the ability to make changes required as dictated by the business.
- Testability, the ability to Validate the software requirements.

Product transition

The product transition perspective identifies quality factors that influence the ability to adapt the software to new environments:-

- Portability, the ability to transfer the software from one environment to another.
- Reusability, the ease of using existing software components in a different context.
- Interoperability, the extent, or ease, to which software components work together.

Product operations

The product operations perspective identifies quality factors that influence the extent to which the software fulfils its specification:-

- Correctness, the functionality matches the specification.
- Reliability, the extent to which the system fails.
- Efficiency, system resource (including cpu, disk, memory, network) usage.
- Integrity, protection from unauthorized access.
- Usability, ease of use.

In total McCall identified the 11 quality factors broken down by the 3 perspectives, as listed above.
For each quality factor McCall defined one or more quality criteria (a way of measurement), in this way an overall quality assessment could be made of a given software product by evaluating the criteria for each factor.
For example the Maintainability quality factor would have criteria of simplicity, conciseness and modularity.

Six Sigma

Six Sigma at many organizations simply means a measure of quality that strives for near perfection. Six Sigma is a disciplined, data-driven approach and methodology for eliminating defects (driving toward six standard deviations between the mean and the nearest specification limit) in any process – from manufacturing to transactional and from product to service.

The statistical representation of Six Sigma describes quantitatively how a process is performing. To achieve Six Sigma, a process must not produce more than 3.4 defects per million opportunities. A Six Sigma defect is defined as anything outside of customer specifications. A Six Sigma opportunity is then the total quantity of chances for a defect. Process sigma can easily be calculated using a Six Sigma calculator.
The fundamental objective of the Six Sigma methodology is the implementation of a measurement-based strategy that focuses on process improvement and variation reduction through the application of Six Sigma improvement projects. This is accomplished through the use of two Six Sigma sub-methodologies: DMAIC and DMADV. The Six Sigma DMAIC process (define, measure, analyze, improve, control) is an improvement system for existing processes falling below specification and looking for incremental improvement. The Six Sigma DMADV process (define, measure, analyze, design, verify) is an improvement system used to develop new processes or products at Six Sigma quality levels. It can also be employed if

a current process requires more than just incremental improvement. Both Six Sigma processes are executed by Six Sigma Green Belts and Six Sigma Black Belts, and are overseen by Six Sigma Master Black Belts.
According to the Six Sigma Academy, Black Belts save companies approximately $230,000 per project and can complete four to six projects per year. (Given that the average Black Belt salary is $80,000 in the United States, that is a fantastic return on investment.) General Electric, one of the most successful companies implementing Six Sigma, has estimated benefits on the order of $10 billion during the first five years of implementation. GE first began Six Sigma in 1995 after Motorola and Allied Signal blazed the Six Sigma trail. Since then, thousands of companies around the world have discovered the far reaching benefits of Six Sigma. Many frameworks exist for implementing the Six Sigma methodology. Six Sigma Consultants all over the world have developed proprietary methodologies for implementing Six Sigma quality, based on the similar change management philosophies and applications of tools.

PARETO ANALYSIS

Pareto Analysis is a statistical technique in decision-making used for the selection of a limited number of tasks that produce significant overall effect. It uses the Pareto Principle (also known as the 80/20 rule) the idea that by doing 20% of the work you can generate 80% of the benefit of doing the entire job. Take quality improvement, for example, a vast majority of problems (80%) are produced by a few key causes (20%). This technique is also called the vital few and the trivial many.

In the late 1940s Romanian-born American engineer and management consultant, Joseph M. Juran suggested the principle and named it after Italian economist Vilfredo Pareto, who observed that 80% of income in Italy went to 20% of the population. Pareto later carried out surveys in some other countries and found to his surprise that a similar distribution applied.

We can apply the 80/20 rule to almost anything:

- 80% of customer complaints arise from 20% of your products and services.
- 80% of delays in the schedule result from 20% of the possible causes of the delays.
- 20% of your products and services account for 80% of your profit.
- 20% of your sales force produces 80% of your company revenues.

- 20% of a systems defects cause 80% of its problems.

The Pareto Principle has many applications in quality control. It is the basis for the Pareto diagram, one of the key tools used in total quality control and Six Sigma.

In PMBOK, Pareto ordering is used to guide corrective action and to help the project team take steps to fix the problems that are causing the greatest number of defects first.

Here are eight steps to identifying the principal causes you should focus on, using Pareto Analysis:

1. Create a vertical bar chart with causes on the x-axis and count (number of occurrences) on the y-axis.
2. Arrange the bar chart in descending order of cause importance that is, the cause with the highest count first.
3. Calculate the cumulative count for each cause in descending order.
4. Calculate the cumulative count percentage for each cause in descending order. Percentage calculation: {Individual Cause Count} / {Total Causes Count}*100
5. Create a second y-axis with percentages descending in increments of 10 from 100% to 0%.
6. Plot the cumulative count percentage of each cause on the x-axis.
7. Join the points to form a curve.
8. Draw a line at 80% on the y-axis running parallel to the x-axis. Then drop the line at the point of intersection with the curve on the x-axis. This point on the x-axis separates the important causes on the left (vital few) from the less important causes on the right (trivial many).

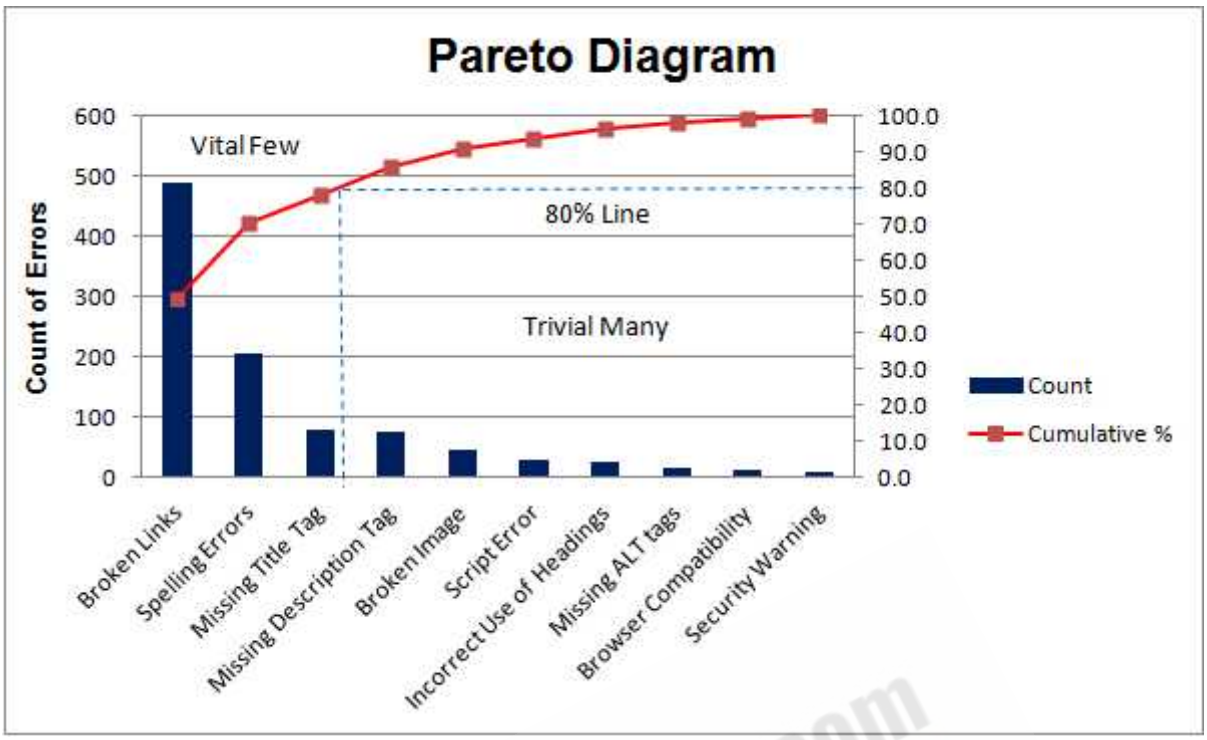


Figure 1: Pareto Analysis Diagram

Here is a simple example of a Pareto diagram, using sample data showing the relative frequency of causes for errors on websites. It enables you to see what 20% of cases are causing 80% of the problems and where efforts should be focussed to achieve the greatest improvement. In this case, we can see that broken links, spelling errors and missing title tags should be the focus.

The value of the Pareto Principle for a project manager is that it reminds you to focus on the 20% of things that matter. Of the things you do for your project, only 20% are crucial. That 20% produces 80% of your results. Identify, and focus on those things first, but don't entirely ignore the remaining 80% of the causes.

Crosby and Striving for Zero defects

Philip Crosby is an American who promoted the phrases “zero defects” and “right first time”. “Zero defects” doesn’t mean mistakes never happen, rather that there is no allowable number of errors built into a product or process and that you get it right first time.

Philip Crosby believes management should take prime responsibility for quality, and workers only follow their managers’ example. He defined the Four Absolutes of Quality Management.

The Four Absolutes of Quality Management

1. Quality is conformance to requirements
2. Quality prevention is preferable to quality inspection
3. Zero defects is the quality performance standard
4. Quality is measured in monetary terms – the price of non-conformance

Crosby's 14 Steps to Quality Improvement

1. Management is committed to quality – and this is clear to all
2. Create quality improvement teams – with (senior) representatives from all departments.
3. Measure processes to determine current and potential quality issues.
4. Calculate the cost of (poor) quality
5. Raise quality awareness of all employees
6. Take action to correct quality issues
7. Monitor progress of quality improvement – establish a zero defects committee.
8. Train supervisors in quality improvement
9. Hold “zero defects” days
10. Encourage employees to create their own quality improvement goals
11. Encourage employee communication with management about obstacles to quality
12. Recognise participants’ effort
13. Create quality councils
14. Do it all over again – quality improvement does not end

Philip Crosby has broadened his approach to include wider improvement ideals. He defined the:

Five characteristics of an“Eternally Successful Organisation”

1. People routinely do things right first time
2. Change is anticipated and used to advantage
3. Growth is consistent and profitable
4. New products and services appear when needed
5. Everyone is happy to work there.

Ishikawa and the Fishbone Diagram

Also Called: Cause–and–Effect Diagram, Ishikawa Diagram

Variations: cause enumeration diagram, process fishbone, time–delay fishbone, CEDAC (cause–and–effect diagram with the addition of cards), desired–result fishbone, reverse fishbone diagram

The fishbone diagram identifies many possible causes for an effect or problem. It can be used to structure a brainstorming session. It immediately sorts ideas into useful categories.

When to Use a Fishbone Diagram

- When identifying possible causes for a problem.
- Especially when a team’s thinking tends to fall into ruts.

Fishbone Diagram Procedure

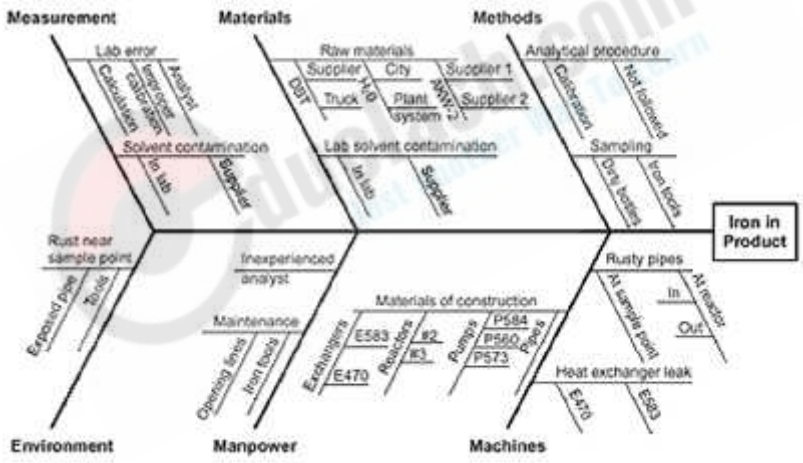
Materials needed: flipchart or whiteboard, marking pens.

1. Agree on a problem statement (effect). Write it at the center right of the flipchart or whiteboard. Draw a box around it and draw a horizontal arrow running to it.
2. Brainstorm the major categories of causes of the problem. If this is difficult use generic headings:
   - Methods
   - Machines (equipment)
   - People (manpower)
   - Materials
   - Measurement
   - Environment

3. Write the categories of causes as branches from the main arrow.
4. Brainstorm all the possible causes of the problem. Ask: “Why does this happen?” As each idea is given, the facilitator writes it as a branch from the appropriate category. Causes can be written in several places if they relate to several categories.
5. Again ask “why does this happen?” about each cause. Write sub–causes branching off the causes. Continue to ask “Why?” and generate deeper levels of causes. Layers of branches indicate causal relationships.
6. When the group runs out of ideas, focus attention to places on the chart where ideas are few.

Fishbone Diagram Example

This fishbone diagram was drawn by a manufacturing team to try to understand the source of periodic iron contamination. The team used the six generic headings to prompt ideas. Layers of branches show thorough thinking about the causes of the problem.



Fishbone Diagram Example

For example, under the heading “Machines,” the idea “materials of construction” shows four kinds of equipment and then several specific machine numbers.

Note that some ideas appear in two different places. “Calibration” shows up under “Methods” as a factor in the analytical procedure, and also under “Measurement” as a cause of lab error. “Iron tools” can be considered a “Methods” problem when taking samples or a “Manpower” problem with maintenance personnel.

Risk Management

Risk management is the identification, assessment, and prioritization of risks (defined in ISO 31000 as the effect of uncertainty on objectives) followed by coordinated and economical application of resources to minimize, monitor, and control the probability and/or impact of unfortunate events or to maximize the realization of opportunities. Risk management's objective is to assure uncertainty does not deflect the endeavor from the business goals.

Risks can come from various sources including uncertainty in financial markets, threats from project failures (at any phase in design, development, production, or sustainment life-cycles), legal liabilities, credit risk, accidents, natural causes and disasters, deliberate attack from an adversary, or events of uncertain or unpredictable root-cause. There are two types of events i.e. negative events can be classified as risks while positive events are classified as opportunities. Several risk management standards have been developed including the Project Management Institute, the National Institute of Standards and Technology, actuarial societies, and ISO standards. Methods, definitions and goals vary widely according to whether the risk management method is in the context of project management, security, engineering, industrial processes, financial portfolios, actuarial assessments, or public health and safety.

Risk sources are identified and located in human factor variables, mental states and decision making as well as infrastructural or technological assets and tangible variables. The interaction between human factors and tangible aspects of risk highlights the need to focus closely on human factors as one of the main drivers for risk management, a "change driver" that comes first of all from the need to know how humans perform in challenging environments and in face of risks (Daniele Trevisani, 2007). As the author describes, «it is an extremely hard task to be able to apply an objective and systematic self-observation, and to make a clear and decisive step from the level of the mere "sensation" that something is going wrong, to the clear understanding of how, when and where to act. The truth of a problem or risk is often obfuscated by wrong or incomplete analyses, fake targets, perceptual illusions, unclear focusing, altered mental states, and lack of good communication and confrontation of risk management solutions with reliable partners. This makes the Human Factor aspect of Risk Management sometimes heavier than its tangible and technological counterpart.

Strategies to manage threats (uncertainties with negative consequences) typically include avoiding the threat, reducing the negative effect or probability of the threat, transferring all or part of the threat to another party, and even retaining some or all of the potential or actual consequences

of a particular threat, and the opposites for opportunities (uncertain future states with benefits).

Certain aspects of many of the risk management standards have come under criticism for having no measurable improvement on risk; whereas the confidence in estimates and decisions seem to increase. For example, it has been shown that one in six IT projects experience cost overruns of 200% on average, and schedule overruns of 70%.

A widely used vocabulary for risk management is defined by ISO Guide 73:2009, "Risk management. Vocabulary.

In ideal risk management, a prioritization process is followed whereby the risks with the greatest loss (or impact) and the greatest probability of occurring are handled first, and risks with lower probability of occurrence and lower loss are handled in descending order. In practice the process of assessing overall risk can be difficult, and balancing resources used to mitigate between risks with a high probability of occurrence but lower loss versus a risk with high loss but lower probability of occurrence can often be mishandled.

Intangible risk management identifies a new type of a risk that has a 100% probability of occurring but is ignored by the organization due to a lack of identification ability. For example, when deficient knowledge is applied to a situation, a knowledge risk materializes. Relationship risk appears when ineffective collaboration occurs. Process-engagement risk may be an issue when ineffective operational procedures are applied. These risks directly reduce the productivity of knowledge workers, decrease cost-effectiveness, profitability, service, quality, reputation, brand value, and earnings quality. Intangible risk management allows risk management to create immediate value from the identification and reduction of risks that reduce productivity.

Risk management also faces difficulties in allocating resources. This is the idea of opportunity cost. Resources spent on risk management could have been spent on more profitable activities. Again, ideal risk management minimizes spending (or manpower or other resources) and also minimizes the negative effects of risks.

According to the definition to the risk, the risk is the possibility that an event will occur and adversely affect the achievement of an objective. Therefore, risk itself has the uncertainty. Risk management such as COSO ERM, can help managers have a good control for their risk. Each company may have different internal control components, which leads to different outcomes. For example, the framework for ERM components includes Internal Environment, Objective Setting, Event Identification, Risk Assessment, Risk Response, Control Activities, Information and Communication, and Monitoring.

Method

For the most part, these methods consist of the following elements, performed, more or less, in the following order.

1. identify, characterize threats
2. assess the vulnerability of critical assets to specific threats
3. determine the risk (i.e. the expected likelihood and consequences of specific types of attacks on specific assets)
4. identify ways to reduce those risks
5. prioritize risk reduction measures based on a strategy

Principles of risk management

The International Organization for Standardization (ISO) identifies the following principles of risk management:

Risk management should:

- create value – resources expended to mitigate risk should be less than the consequence of inaction
- be an integral part of organizational processes
- be part of decision making process
- explicitly address uncertainty and assumptions
- be a systematic and structured process
- be based on the best available information
- be tailorable
- take human factors into account
- be transparent and inclusive
- be dynamic, iterative and responsive to change
- be capable of continual improvement and enhancement
- be continually or periodically re-assessed

Change Management:

Change management is the discipline that guides how we prepare, equip and support individuals to successfully adopt change in order to drive organizational success and outcomes.

While all changes are unique and all individuals are unique, decades of research shows there are actions we can take to influence people in their individual transitions. Change management provides a structured approach for supporting the individuals in your organization to move from their own current states to their own future states.

THREE LEVELS OF CHANGE MANAGEMENT

Individual Change Management

While it is the natural psychological and physiological reaction of humans to resist change, we are actually quite resilient creatures. When supported through times of change, we can be wonderfully adaptive and successful.

Individual change management requires understanding how people experience change and what they need to change successfully. It also requires knowing what will help people make a successful transition: what messages do people need to hear when and from whom, when the optimal time to teach someone a new skill is, how to coach people to demonstrate new behaviors, and what makes changes “stick” in someone’s work. Individual change management draws on disciplines like psychology and neuroscience to apply actionable frameworks to individual change.

Organizational/Initiative Change Management

While change happens at the individual level, it is often impossible for a project team to manage change on a person-by-person basis. Organizational or initiative change management provides us with the steps and actions to take at the project level to support the hundreds or thousands of individuals who are impacted by a project.

Organizational change management involves first identifying the groups and people who will need to change as the result of the project, and in what ways they will need to change. Organizational change management then involves creating a customized plan for ensuring impacted employees receive the awareness, leadership, coaching, and training they need in order

to change successfully. Driving successful individual transitions should be the central focus of the activities in organizational change management.

Organizational change management is complementary to your project management. Project management ensures your project's solution is designed, developed and delivered, while change management ensures your project's solution is effectively embraced, adopted and used.

Enterprise Change Management Capability

Enterprise change management is an organizational core competency that provides competitive differentiation and the ability to effectively adapt to the ever-changing world. An enterprise change management capability means effective change management is embedded into your organization's roles, structures, processes, projects and leadership competencies. Change management processes are consistently and effectively applied to initiatives, leaders have the skills to guide their teams through change, and employees know what to ask for in order to be successful.

The end result of an enterprise change management capability is that individuals embrace change more quickly and effectively, and organizations are able to respond quickly to market changes, embrace strategic initiatives, and adopt new technology more quickly and with less productivity impact. This capability does not happen by chance, however, and requires a strategic approach to embed change management across an organization.

Software Reliability

Software Reliability is the probability of failure-free software operation for a specified period of time in a specified environment. Software Reliability is also an important factor affecting system reliability. It differs from hardware reliability in that it reflects the design perfection, rather than manufacturing perfection. The high complexity of software is the major contributing factor of Software Reliability problems. Software Reliability is not a function of time - although researchers have come up with models relating the two. The modeling technique for Software Reliability is reaching its prosperity, but before using the technique, we must carefully select the appropriate model that can best suit our case. Measurement in software is still in its infancy. No good quantitative methods have been developed to represent Software Reliability without excessive limitations. Various approaches can be used to improve the reliability of software, however, it is hard to balance development time and budget with software reliability.

Key Concepts

Definition

According to ANSI, Software Reliability is defined as: the probability of failure-free software operation for a specified period of time in a specified environment. [Although Software Reliability is defined as a probabilistic function, and comes with the notion of time, we must note that, different from traditional Hardware Reliability, Software Reliability is not a direct function of time. Electronic and mechanical parts may become "old" and wear out with time and usage, but software will not rust or wear-out during its life cycle. Software will not change over time unless intentionally changed or upgraded.

Software Reliability is an important to attribute of software quality, together with functionality, usability, performance, serviceability, capability, installability, maintainability, and documentation. Software Reliability is hard to achieve, because the complexity of software tends to be high. While any system with a high degree of complexity, including software, will be hard to reach a certain level of reliability, system developers tend to push complexity into the software layer, with the rapid growth of system size and ease of doing so by upgrading the software. For example, large next-generation aircraft will have over one million source lines of software on-board; next-generation air traffic control systems will contain between one and two million lines; the upcoming international Space Station will have over two million lines on-board and over ten million lines of ground support software; several major life-critical defense systems will have over five million source lines of software. While the complexity of software is inversely related to software reliability, it is directly related to other important factors in software quality, especially functionality, capability, etc. Emphasizing these features will tend to add more complexity to software.

Software failure mechanisms

Software failures may be due to errors, ambiguities, oversights or misinterpretation of the specification that the software is supposed to satisfy, carelessness or incompetence in writing code, inadequate testing, incorrect or unexpected usage of the software or other unforeseen problems. While it is tempting to draw an analogy between Software Reliability and Hardware Reliability, software and hardware have basic differences that make them different in failure mechanisms. Hardware faults are mostly physical faults, while software faults are design faults, which are harder to visualize, classify, detect, and correct. Design faults are closely related to fuzzy human factors and the design process, which we don’t have

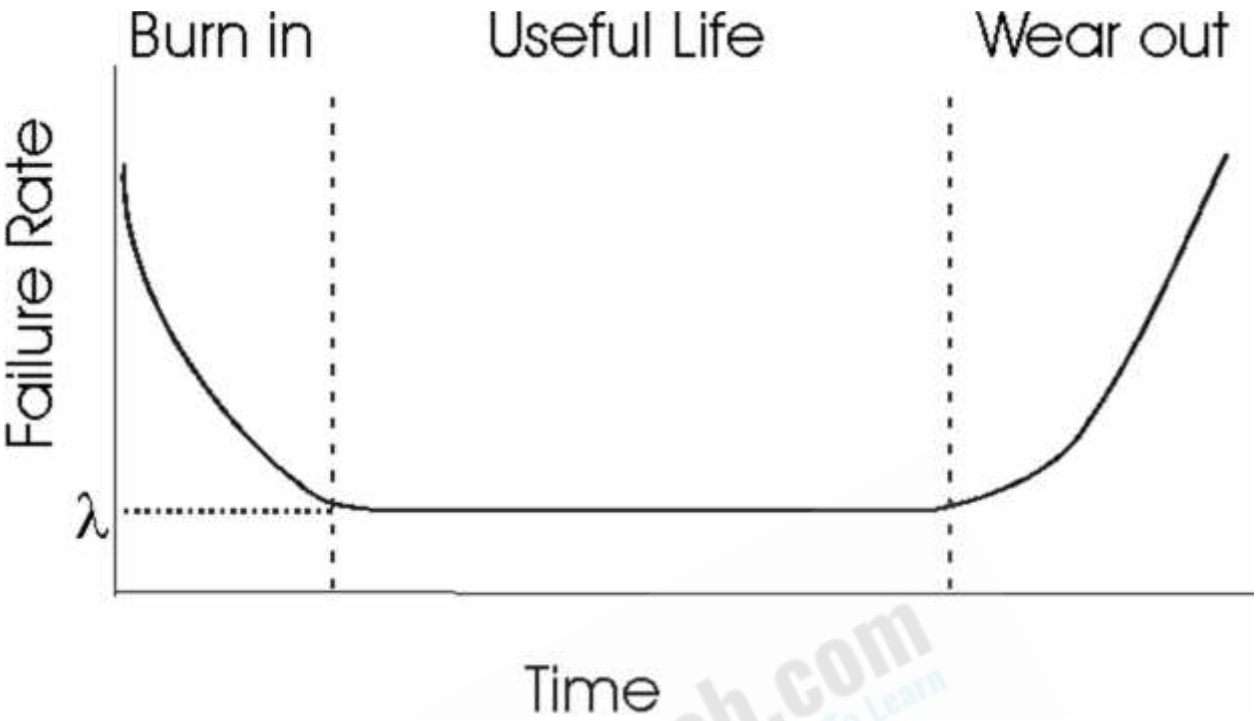
a solid understanding. In hardware, design faults may also exist, but physical faults usually dominate. In software, we can hardly find a strict corresponding counterpart for "manufacturing" as hardware manufacturing process, if the simple action of uploading software modules into place does not count. Therefore, the quality of software will not change once it is uploaded into the storage and start running. Trying to achieve higher reliability by simply duplicating the same software modules will not work, because design faults cannot be masked off by voting.

A partial list of the distinct characteristics of software compared to hardware is listed below:

- Failure cause: Software defects are mainly design defects.
- Wear-out: Software does not have energy related wear-out phase. Errors can occur without warning.
- Repairable system concept: Periodic restarts can help fix software problems.
- Time dependency and life cycle: Software reliability is not a function of operational time.
- Environmental factors: Do not affect Software reliability, except it might affect program inputs.
- Reliability prediction: Software reliability cannot be predicted from any physical basis, since it depends completely on human factors in design.
- Redundancy: Cannot improve Software reliability if identical software components are used.
- Interfaces: Software interfaces are purely conceptual other than visual.
- Failure rate motivators: Usually not predictable from analyses of separate statements.
- Built with standard components: Well-understood and extensively-tested standard parts will help improve maintainability and reliability. But in software industry, we have not observed this trend. Code reuse has been around for some time, but to a very limited extent. Strictly speaking there are no standard parts for software, except some standardized logic structures.

The bathtub curve for Software Reliability

Over time, hardware exhibits the failure characteristics shown in Figure 1, known as the bathtub curve. Period A, B and C stands for burn-in phase, useful life phase and end-of-life phase. A detailed discussion about the curve can be found in the topic Traditional Reliability.


Burn in
Useful Life
Wear out
Failure Rate
λ
Time