Prof:Siddhesh Zele's

PROJECT MANAGEMENT

TYBSC(IT) SEM 6

COMPLIED BY : MANDAR GHAG AND SIDDHESH ZELE

302 PARANJPE UDYOG BHAVAN, NEAR KHANDELWAL SWEETS, NEAR THANE STATION , THANE (WEST)

PHONE NO: 8097071144 / 8097071155 / 8655081002

UNIT I

1. Conventional Software Management

Syllabus:

Conventional Software Management: The Waterfall model, Conventional software management performance

1.1 Introduction

Software management is a process through which a software is managed throughout its development process.

The software management also called as *software project management (SPM)* deals with managing the critical aspects right from start of the planning of the project to deployment and maintenance.

SPM manages the following aspects of software development;

- version tracking,
- developing,
- testing,
- integration,
- configuring,
- installing, and
- distributing

In short, SPM is about managing:

- software engineering work
- encouraging the stakeholder interaction in the early SDLC stages so as to avoid
- paying attention to the technical tools and methods used in the process
- planning the product objectives and scope, cost

5PM specially focuses on four P's:

1. People
2. Product
3. Process
4. Project

1.2 Conventional Software Management

As per the analysis of the software engineering industries of mid 1990s, it yielded that the Software Development was highly unpredictable because:

- Only about 10% of software projects were delivered successfully in time, in budget, and meeting the user requirements.
- The management discipline acted more as a discriminator in success or failure or when in case of technology advances.
- The level of software scrap and rework was indicated to be at an immature level process. The reasons behind these analyzation may either be bad theory or bad practice or both.

1.3 The Waterfall Model

We have already learnt about waterfall model in 'software engineering' subject. A brief overview of this model is shown in Fig. 1.3.1;

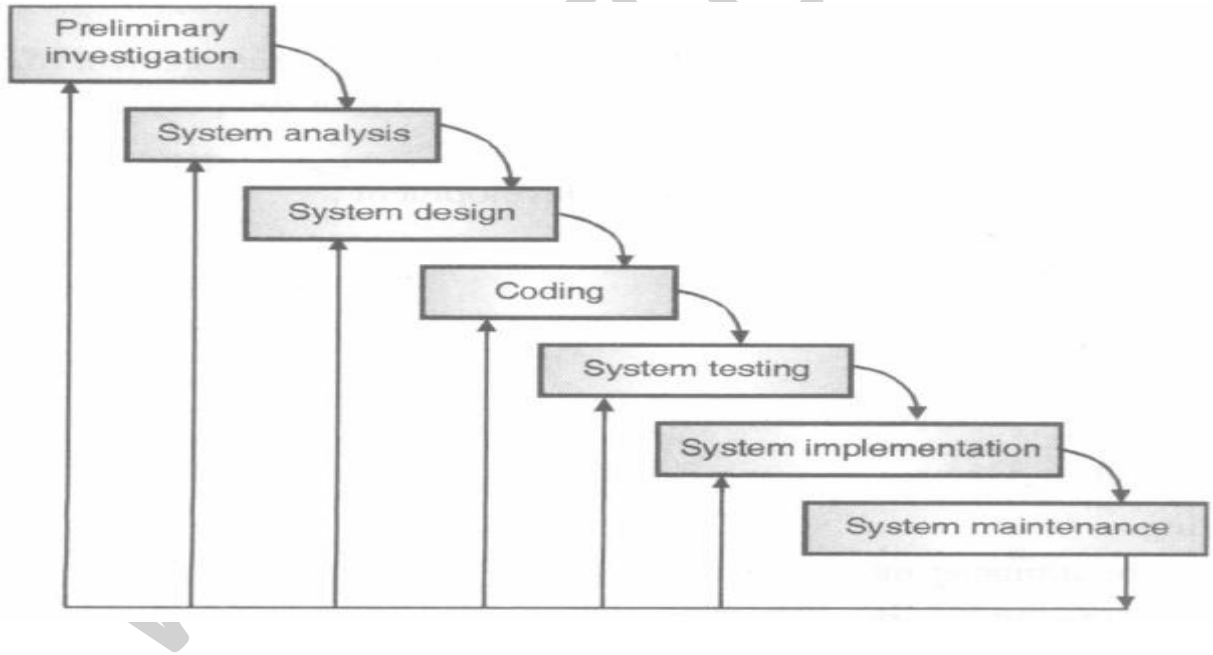


Fig. 1.3.1: Waterfall model

The detail description about the working of this model is not necessary, here. Therefore, we will only see the various perspectives of the 'waterfall model' in earlier days (historical perspective) and how it is updated nowadays. This linear waterfall model was first proposed by 'Winston Royce'. It suggests systematic sequential approach for software development. It is the oldest paradigm for software engineering i.e. it is the oldest software development life cycle.1.3.1 Preliminary Investigation :

Preliminary investigation means total inspection of the existing system i.e. clear understanding of the system. Its basic task is to find out real problem of the system with causes and complexity of the problem. Its secondary but very important task is to find out all possible solutions to solve that problem and according to that which solution is feasible for the system in terms of technology ,cost, operational. Its last task is to mention all benefits can be expected after problem is solved.

So this phase is divided into three main goals as follows :

- Problem identification
- Possible and feasible problem solution i.e. Feasibility study.
- Expected benefits after the problems are solved.

1.3.1.1 Problem Identification (Problem Analysis):

It requires to completely investigate the environment of the system. Generally it requires studying two environments - Internal environment and external environment, which are listed below :

Sr. No.	Internal Environment	External Environment
1	Company Management	Customers
2	Employees of all departments	Management consultant
3	Internal auditors	External auditors
4	Data Processing department	Government Policies
5	Financial Reports	Competitions

There are normally seven types of problems encountered in the system :

- ***Problem of Reliability*** *:* If system may not work properly or same procedures give different (i.e. unreliable) result.
- ***Problem of Validity*** *:* Reports contain misleading information.
- ***Problem of Economy*** *:* System is costly to maintain.
- ***Problem of Accuracy*** *:* Reports have many errors.
- ***Problem of Timeliness*** : Every work requires large time.
- ***Problem of Capacity*** : Capacity of the system is inadequate.
- ***Problem of Throughput*** : System does not produce expected results, or we can say system has more capacity but it accomplishes very less work as compared to capacity. The main *advantage*

of waterfall model is, it exactly pin points the problem. So it is very useful in setting all goals of the system as well as used to decide system boundaries.

1.3.1.2 Project Feasibility Study :

Feasibility study is essential to evaluate cost and benefit of the proposed system. This is very important step because on the basis of this; system decision is taken on whether to proceed or to postpone the project or to cancel the project.

Need of Feasibility Study :

- It determines the potential of existing system.
- It finds or defines all problems of existing system.
- It determines all goals of the system.
- It finds all possible solutions of the problems of existing system. We can call it as proposed system.
- It finds technology required to solve these problems.
- It determines really which solution is easy for operational from the point of view of customer and employees such that it requires very less time with 100% accuracy.
- It determines what hardware and software is required to obtain solution of each problem or proposed system.
- It determines cost requirements of the complete proposed system in terms of cost of hardware required, software required, designing new system, implementation and training, proposed maintenance cost.
- It avoids costly repairs, crash implementation of new system.
- It chooses such system which is easy for customer to understand and use so that no special training is required to train the customer. It may give some training to employees of the system.

Method :

Steering committee :

This committee conducts detailed study. This committee first studies existing system and identifies all problems and looks into three types of feasibility study. Those are given below :

- Technical feasibility
- Operational feasibility
- Economical feasibility
- Organizational feasibility
- Cultural feasibility

Technical Feasibility :

- The committee first finds out *technical feasibility of the existing system.* It involves following steps :
- It determines available hardware.
- It determines available computer with configuration.
- It determines available software.
- It determines operating time of available system that is computer, hardware software.

After that it finds out technical feasibility required for the proposed system. It involvesfollowing steps:

- It mentions new hardware requirements of proposed system.
- It mentions computer with new configuration requirement of proposed system.
- It mentions new software requirements of proposed system.
- It mentions new operating time of available system that is computer, hardware,software.
- It mentions old as well as new facilities which will be provided by the proposed system.
- It mentions all benefits of the system.

Operational Feasibility :

It is also called as behavioral feasibility. It finds out whether the new technology or proposed system will be suitable using three type of aspects; that are human, organizational and political aspects. It involves following steps :

1. It finds out whether there is any direct-indirect resistance from the user of this system or not ?
2. It finds whether the operation of proposed system is easy or not as compare to existing system.
3. It finds out whether the user or customer of the system requires extra training or not ?
4. If it requires any retraining then it is accepted by user as well as customer or not ?
5. It finds if any job reconstruction is required or not ?
6. It finds if this reconstruction of the job is accepted in organization ?
7. It also finds if it is acceptable then what should be the skill sets of that job.
8. Watches the feelings of the customers as well as users.
9. It should provide right and accurate information to user or customers at right place as well as at right time.

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

Economical Feasibility :

Here, steering committee finds total cost and all benefits as well as expected savings of the proposed system.

There are two types of costs - ***onetime cost*** **and** ***recurring costs***. **One time cost** involves following :

1. Feasibility study cost.
2. Cost converting existing system to proposed system.
3. Cost to remolding architecture of the office, machineries, rooms etc.
4. Cost of hardware's.
5. Cost of Operating system software.
6. Cost of Application software.
7. Technical experts consulting costs.
8. Cost of training.
9. Cost of Documentation preparation

Recurring cost involves following :

1. Cost involves in purchase or rental of equipment.
2. Cost of phones and mobiles Communication equipment.
3. Cost of Personnel search, hiring, staffing.
4. Cost of Salaries of employee's.
5. Cost of supplier's.
6. Cost of maintenance of equipment.

Organizational Feasibility :

This demonstrates the management capability and availability, employee involvement, and their commitment. This shows the management and organizational structure of the project, ensuring that the system structure is as described in the requirement analysis or SRS and is well suited to the type of operation undertaken.

Some organizational risks impacting new system :

- Low level of computer competency among employees
- Perceived shifting of organizational power
- Fear of employment loss due to increased automation
- Reversal of long-standing work procedures

One way to overcome these risks : training sessions.

Cultural Feasibility :

- It deals with the compatibility of the proposed new project with the cultural environment of the project.
- In labor-intensive projects, planned functions must be integrated with the local cultural practices and beliefs.

Example :Religious beliefs may influence what an individual is willing to do or not do.

Example :An enterprise's own culture can clash with the results of the project.

In these conditions, the project's alternatives are evaluated for their impact on the local and general culture.

1.3.2 System Analysis :

This phase is about complete understanding of all important facts of the business based on the preliminary investigation. It involves following activities :

1. It is the study of all components of the system as well as inter relation between all components of the system and relation between components and environment.
2. It determines what is to be done in the organization.
3. It finds the procedures of how to do that.
4. What is the input data ?
5. What is the procedure through which inputs are to be converted into output ?
6. When should the transactions occur on the data?
7. When problems arise, determine the solutions to solve it and what are the reasons behind those problems.

Methodology of System Analysis :

1. Identifying the system boundary
2. Understanding the role and interrelationship of elements with other elements of the same system.

The above two methodologies generate:

the capability to analyze and compare various alternatives regarding components and system functioning and system objectives

Need /Advantages of System Analysis :

- provides greater understanding of the complex structures
- it acts as a tradeoff between functional requirements of subsystems to the total system
- it helps in understanding and comparing the functional impacts of subsystems to the total system

- it helps in achieving the inter-compatibility and unity of purpose of sub systems
- helps in discovering means to design systems
- helps in placing each subsystem in its proper place and context, so that the system as a whole may achieve its objectives with minimum available resources.

Objectives of System Analysis :

1. Define the system.
2. Divide the system into smaller parts.
3. Finds all nature, function and inter-relationship of various parts of the systems.

If the system is not analyzed properly then there may be problem in the Preliminary investigation phase.

1.3.3 Software Design :

The main objective of this phase is to design proposed system using all information collected during preliminary investigation and directed by the system analyst. This is a challenging phase and includes following steps :

1. Design of all types of inputs of proposed system.
2. Design of all types of outputs of proposed system.
3. Design of the procedures which convert input to output.
4. Design of the flow of information.
5. Design of the information which is required to store within a files and data bases Volumes.
6. Design of collection of inputs using forms (Manual forms).
7. Design in terms of program specification i.e. logical design.
8. It determines the hardware cost, hardware capability.
9. It determines the speed of software.
10. It determines error rates, and other performance characteristics are also specified.
11. It also considers the changes to be made in the organizational structure of the firm in design.
12. This phase also designs standards for testing, documentation.

Generally traditional tools are used for the designing of the procedures that are as follows :

- Flowcharts
- Algorithms
- IPO (i.e. Input Processing and output) and HIPO (Hierarchy of IPO) charts
- Decision tables
- Data Flow Diagrams

1. If system design phase is facing problem during the design then first go back to the system analysis phase and redesign the system but if problem is not solved then go for preliminary investigation.

2. If System design phase produces all expected results then it goes to next phase.

1.3.4 Coding:

This phase is just implementing the design in to programming language that means it actually develops the proposed system. It involves the following steps :

1. It first of all, finds out the best suitable programming language that is suitable for the design as well as also suitable in the organization.
2. It accepts design and break system modules into smaller programs.
3. It develops or writes program for each part in selected programming language.
4. Prepares documentation that means add necessary comment lines wherever necessary within a program.
5. Now it combines all small programs together and builds one big program.
6. If any problem occurs during the coding phase then waterfall model tries to solve it by repeating system design phase :

- If that problem does not get solved then waterfall model repeats system analysis phase and system design phase.
- If that problem does not get solved then waterfall model repeats from first phase i.e. preliminary phase through system analysis phase and system design phase.

7. If coding phase produces all expected results then it goes to next phase.

1.3.5 Testing :

This phase includes the testing of the code or programs developed by the coding phase. This includes following steps :

1. First of all, it finds out all possible expected results (i.e. output data) for the set of input data.
2. It also checks the validity of the input data as well as checks expected output data.
3. It finds out all wrong results and immediately tries to correct it by repeating coding phase.
4. It finds the speed of functions using special codes.
5. It determines whether each program can perform the intended tasks or not?
6. It checks result by test data.
7. It checks the logic of the individual programs.
8. It checks interfaces between various programs.
9. It checks quality of code in terms of speed and space.
10. It checks whether system has produced correct and desired results which lead to designated goals.
11. If testing finds out that the system does not produce expected result then this problem is solved by repeating the previous phases as needed.
12. If testing phase finds out that the system produces all expected results then it goes to next phase.

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

1.3.6 System Implementation :

System Implementation is not creative process but it is somewhat difficult task. This phase has two parts - *implementation* and *evaluation* of the system.

Implementation :

There are two ways of implementation. Those are as follows :

1. Implement proposed system with existing old system and find out performance of the both systems and slowly replace new system with older one.
2. Totally replace old system with proposed new system.

Risk factor of second type of implementation is more as compare to first one. Second step needs strict evaluation.

Both types of implementation consist of following steps :

1. It prepares site for new proposed system.
2. It installs required hardware within a system.
3. It installs required software in a system.
4. It installs a developed code i.e. programs in a system.
5. It prepares training program for user of the proposed system as well as customers of the system.
6. It prepares user manual which includes all the steps which give guidance to user.
7. It gives training to all types of users of proposed system.
8. Observe the system when users of system are using it.
9. If users are facing any problems regarding the new system, it tries to find out exact phase from where root cause of the problem starts, and accordingly starts waterfall model.

Evaluation :

Evaluation is nothing but feedback for the system. It is very essential check point of the system which is the process of verifying the capability of a system. It continuously evaluates and checks whether proposed system is meeting the objectives or not. It includes :

1. Development Evaluation :

- It checks whether the system is developed within time.
- It checks whether the system is developed within the budget.
- System is passed by development methods and tools.

2. Operational Evaluation :

- It checks response time of proposed system.

- It checks whether it is really easy to use or not?
- It checks accuracy of computations (It is seen in testing also).
- It checks storage capacity.
- It checks reliability.
- It checks functioning of the existing system.
- Collects necessary feedback from users.
- It finds all benefits of the proposed system.
- Collects information of attitude of different persons regarding proposed system.
- It evaluates cost, time and effort taken for the overall project.

1.3.7 Maintenance:

Maintenance is the process in which it finds out essential changes (i.e. new trends) of the market or business to correct some errors and tries to implement it in the existing system.

There are usually three types of maintenance, that are :

1. **Correction :**

Sometimes, proposed system has few types of errors; and it is the duty of software engineer to correct it as soon as it is encountered by the user. Generally there are four types of errors, that are as follows :

- Minor changes in the processing logic.
- Errors detected during the processing.
- Revisions of the formats for data inputs.
- Revisions of the formats of the reports.

These errors can be corrected by repeating waterfall model from coding phase through testing, implementation and maintenance.

2. **Adaptation :**

Sometimes, our proposed system is executable on Windows environment, but somebody wants to run it in LINUX environment, or some other operating system. Then we are required to design our proposed system from third phase that is from System Design phase.

3. **Enhancement :**

Because of new technology and business competition, organization needs to imply or to add new functions or additional capabilities to the proposed system.

After some time, people think that some techniques may be used in the system so some additional features can also be added into it. Sometimes, new hardware is required to add some extra features. For enhancement it may repeat whole system or may repeat it from design phase or sometimes from coding phase.

Advantages of Waterfall Model:

1. It defines very first software development process.
2. The product of waterfall model always defines all constraints of the organization.
3. It always produces a good quality product in terms of space and time.

Drawbacks of the Waterfall Model are as follows :

1. Real products rarely follow this sequential flow.
2. Because of iteration, changes can cause confusion as the project team proceeds
3. It is very difficult for customer to state all the requirements in one time.
4. Many projects face this uncertainty at beginning only, so it is very difficult to design next phases.
5. Time span required for each phase could not be specified.
6. Naturally project requires more time.
7. Project becomes lengthy also.
8. Customer should have patience.

It tries to solve time consumption in early stages.

1.4 Historical Perspective

1. In earlier days, developers gave more importance to the two most essential steps i.e. 'analyses' and 'coding'. These two steps were interpreted as the two most important steps at that time.
2. So as to manage and control the activities of the software development process, the developers introduced various 'overhead' steps such as the system requirements definition, software requirements definition, program design, and testing. All these steps supported in enhancing the analysis and coding steps."
3. The conventional basic waterfall model is risky and failure prone. This is because as the testing phase occurs at the end of the development life cycle, and as a result, the changes identified are likely to be so troublesome to implement right from the software requirements on which the design is based.

1.4.1 Suggested Changes

1. *Earlier,"* **Program design comes first " .**

- Mostly, Designing was done between SRS and analysis phases.
- Program designer first checks the storage, timing, and data.

- During the Analysis phase, the Designer imposes timing and operational constraints so as to cross-check the consequences.
- The software development design process is always built by the program designers, and not by the analysts. The program designers define and allot various data processing modes (such as allocating functions, designing databases, interfacing, processing modes (such as the i/o processors), operating procedures(such as the one entering into a branch even when it is wrong.
- Training is give to the staff until they learn and use the software

Now; **"Architecture comes first "** is given importance and comes first rather than program design.

That means, nowadays, the basic ARCHITECTURE comes FIRST.

- It includes the elaboration of architecture by distributing the system into components, and representing these components in a layered architecture.
- For example, the Rational Unified process (RUP) includes use-case driven, architecture-centric, iterative development process.
- These architectures are THEN designed and developed in parallel with planning and requirements definition.

2. *Earlier;* **" Program design comes first "**THEN, document these Designs

- In earlier days of software development, the Development required huge amounts of documentation i.e. everything had to be represented using manuals.
- These manuals are called as User manuals. These included operational manuals, software maintenance manuals, staff user manuals, test manuals and etc.
- Maintaining such a huge amount of documentation becomes really troublesome.
- It was MUST for each designer to communicate with various stakeholders such as interface designers, managers, customers, testers, developers so as to finalize the designs and prepare the manuals.

Now, **" Document the Design first"** THEN derive the actual designs

- Nowadays, 'artifacts' are given primary importance. These models(documentation / artifacts) are derived from the developed architecture, analysis report, captured requirements, and these models (documentation / artifacts) are then used in deriving a design solution.
- This documentation /artifacts include Use Cases, static models such as class diagrams, state diagrams, activity diagrams, dynamic models such as sequence and collaboration diagrams, domain models, glossaries, supplementary specifications (such as constraints, operational environmental constraints, distribution constraints, etc.)
- Modem designing tools and notations, and new methods produce self documenting artifacts from development activities. For example, Rational Rose is a very popular design tool that produces the self-documenting artifacts and program code from the developed model.

- Therefore, we can say that, Visual modeling provides significant documentation.

3. *Earlier;* "Do it twice" THEN, document these Designs

- It is sometimes very confusing when we do the same program twice i.e. the first version of the program may contain some errors or say, some loop holes, so we make changes in it and derive it's second version. But, later on, keeping track of the most recent version of these programs become troublesome.
- Version 1 has major problems and therefore, alternatives are addressed – for example, the changes are done in 'big cookies' such as communications, user interfaces, data models, hardware/software platforms, operational or any other constraints. Therefore, there is a need to track the correct version of the program and also, if needed the first version is thrown away sometimes.
- Then version 2 is called as the refinement of version 1 which includes the implementation of major requirements.
- *Now;* "Architecture -first Development" THEN, derive the actual designs
- This approach is the base to architecture-first development. This is known as 'Initial engineering' which forms the basis for *Iterative development* and also helps in addressing the risks.

4. *Earlier;* "Plan , Control, and Monitor Testing"

- Testing phase utilizes the greatest number of project resources which includes manpower, computation time, ultimately cost and schedule.
- Therefore, it involves greatest risks also in terms of cost and schedule.
- The testing phase occurs in the last of software development and this last phase is reached when any other changes or any alternatives are least available and expenses are at a maximum.
- This phase involves :

1. Employing a team of test specialists and these specialists not involved in the original design and development of the software product.
2. Employing visual inspections to detect the obvious errors. These inspections include the procedure of code reviews, technical reviews and interfaces.
3. Testing every logical path
4. Final check out on the customer's computer.

Now; " Plan , Control, and Monitor Testing"

- The 1st and 4th testing activities are still valid i.e.;

1. Employing a team of test specialists and these specialists not involved in the original design and development of the software product.
4. Final checkout on the customer's computer.

- But the 2nd and 3rd activities differ from the earlier perspectives;

2. Software inspections - are conducted by automated tools but they are assisted by the code analyzers, optimizing compilers, static and dynamic analyzers.
3. Testing every path - This is practically impossible. It is especially very difficult with distributed systems and reusable components and there are even many other factors that need to be tested.

5. *Earlier;* "Involvement of the Customers" .

- It involves customers' participation in requirements' definition, in preliminary software review, and also in preliminary program design.
  Now, " Involvement of the Customers"
- It involves the customers also and all other stakeholders also. This involvement is necessary for the overall project success.
- It demonstrates the increments, the customer feedback, making favorable changes, cyclic and iterative activities that yield in an evolving software.
- Involvement of customers helps in addressing the risks in very initial phases of software development.

1.5 The Software Development Plan

If you observe the *Old Versions* of Software Development Plan, you will see that the success rate is less than 20%.

The old versions of software development plan defines :

- precise requirements
- precise plan to deliver and deploy the system constrained by specific time and budget.
- Executes and tracks to plan

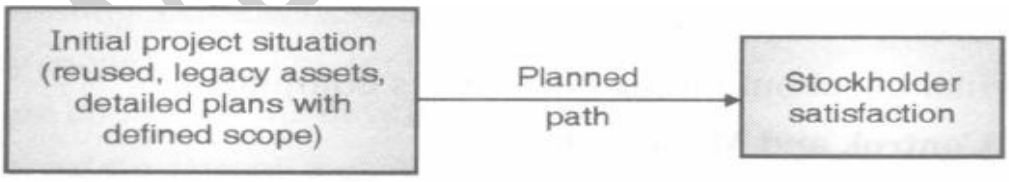


Fig. 1.5.1 : Old Version of Software Development

- In practice, these activities of software development plan didn't help much in increasing the success rate.

- Projects are not delivered on-time, not within planned budget, and rarely met the user requirements.
- Projects that are developed using Conventional waterfall process often faced following symptoms:

1. Protracted integration and late design breakage
2. Late risk resolution
3. Requirements-driven functional decomposition
4. Adversarial stakeholder relationships
5. Focus on documents and review meetings

1.5.1 Protracted (Prolonged/Late) Integration and Late Design Breakage

- The activities that are carried out in conventional waterfall process are listed down in a sequential manner: Requirements - Design - Code - Integration –

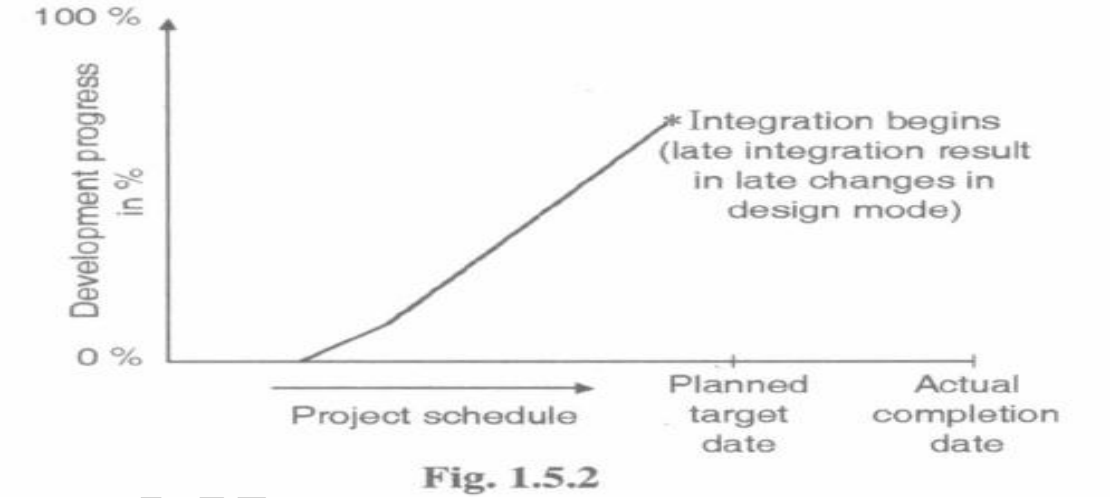


Fig. 1.5.2

the integration of components very late in the SDLC, then it results in late changes in the designed models. This is known as late design i.e. coming back to design phase when you are at the last phases of the life cycle. This late designing causes the breakages in the developed product.

- The late integration test and designing also increases the expenditures of the software project.

Table 1.5.1: Expenditures per activity for a Conventional Software Project

Activity		Cost
Management	–	5%
Requirements	–	5%
Design	–	10%
Code and unit test	–	30%
Integration and Test	–	40%
Deployment	–	5%
Environment	–	5%
Total	–	**100%**

- Lot of time is spent on *finalizing the software design* and then after spending so much of time on designs and after perfecting them to their satisfaction, then the task of coding is started.
- Generally, the requirements are noted down in English, designs are drawn using flowcharts, detailed designing is done in pal, and implementations are basically done in Fortran, Cobol, or C.
- The problems of 'Waterfall model - late integration' that effect the software performance are
  - Only unit testing is performed from the start of coding but all other tests could be performed only 'at the end' of the SDLC phases,
  - Thus, the late integration and testing phase consumes 40% of life-cycle resources: which will not if we start the integration of components and testing at the right time.

1.5.2 Late Risk Resolution

- Focuses on early prepared paper artifacts.
- Actual risks and issues are still unknown and difficult to detect and understand.
- The below figure shows a sample risk profile for projects that follow waterfall model. It involves four different periods of risk exposure;

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

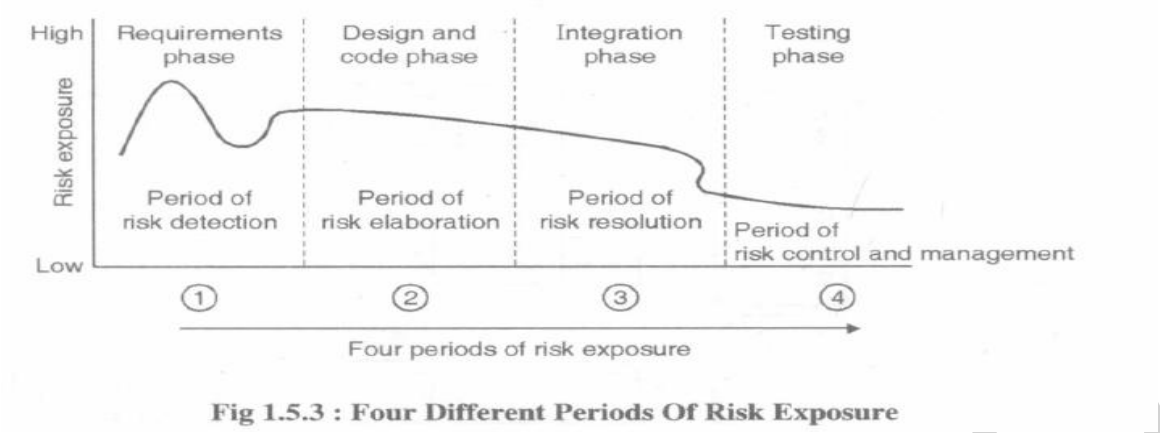


Fig 1.5.3 : Four Different Periods Of Risk Exposure

- Difficult to identify and resolve the risks right during the requirement gathering because this is the initial phase of SDLC and in this phase, many key items are still not fully understood.
- Even in the design phase, where we are clear enough with the software requirements and have better understood them, then also it is still difficult to analyze and finalize the objectives.
- During the coding phase, some of the risks are resolved, but more than that, especially at the time of integration, many of the risks become quite clear and accordingly changes are made to the artifacts and reduction in expenditures can be achieved.
- Even if we achieve success in reducing the expenditures, it often results in extending the scheduled dates of delivering the product.
- This may also hamper the quality a little bit due to frequent changes and extensibility, or in the process of maintainability. Thus, it ultimately results in the loss of original design and integrity.

1.5.3 Requirements-driven Functional Decomposition

- From the traditional perspective, the software development processes have been interpreted as requirements-driven as shown in the below Fig. 1.5.4.
- Developers must focus on gathering and writing complete, clear, precise, consistent, necessary, and feasible user requirements. But this case was rarely observed.
- Very often, it is seen that too much time is spent on treating *all* requirements (such as the requirements that are normally listed in. condition-action tables, decision-logic tables, flowcharts, or plain text) *equally* rather than spending more time on critical ones.

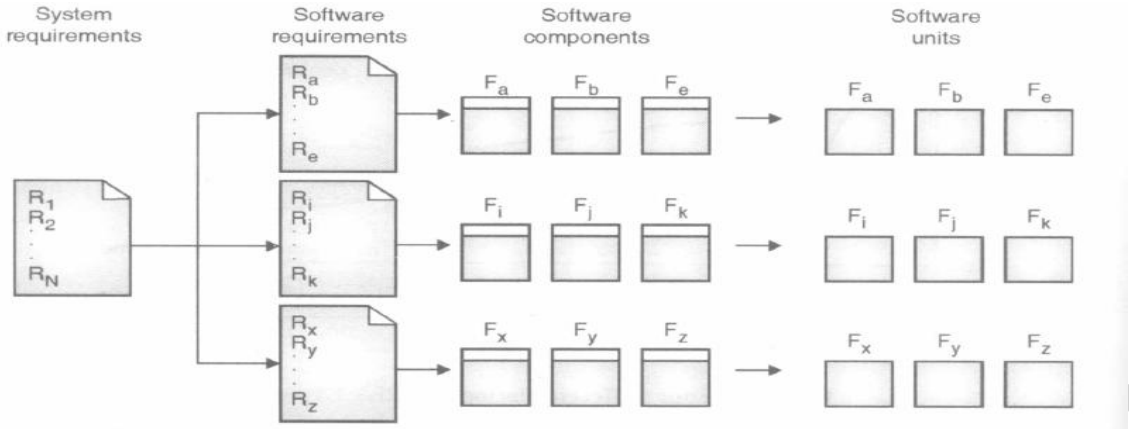


Fig. 1.5.4 : Component Organization interpreted from A Requirements-Driven Approach

- Much of the brainpower and time is wasted on the 'less important' requirements. That means, in traditional way of development, prioritizing of requirements was given least importance.
- Also, much time is spent on documentation i.e. documenting the features (traceability, testability, etc.) of the software which were later made obsolete as the requirements and designs subsequently evolve.
- Also, there is a false assumption in conventional software development that all the requirements can be implemented as 'functions' in the coding phase and this assumption leads to the decomposition of these functions.
- This conversion of requirements into functions and then leading to decomposition of Functions into sub-functions, etc. has become the basis for contracts and work distribution, while ignoring the major architectural-driven approaches that are *threaded* throughout the functions and that surpass every individual functions such as the security; authentication; persistency; and performance.

2.5.4 Adversarial Stakeholder Relationships

- First of all, there is a need to decide 'who are stakeholders'?
- Lot of misunderstandings occur between the stakeholders about the matter written in the documentation because of the English jargons.
- Requirements are only documented on papers but in actual practice, they are not really modeled.
- Universally-agreed notations are rarely used, no common notations, no GUIs.
- Subjective reviews and different stakeholder opinions are not given enough value.

Few common events that take place in contractual software are:

1. Contractor prepared a draft that constitutes of contract-deliverable document that involves *intermediate artifacts* and delivered it to the customer for approval. Thisis usually done after interviews, questionnaires, and meetings.
2. Customer was ought to give feedback within 15-30 days.
3. Contractor incorporated this feedback and submitted within 15-30 days. This is interpreted as the final version for approval.

Observation :

- Huge paper work such that it became 'intolerable' and often some of the documentation was under-read or un-read.
- Tense contractor/customer relationships
- Mutual distrust which was the basis for much of the problems.
- It was often seen that, once approved is rendered obsolete later.

1.5.5 Focus on Documents and Review Meetings

- It is a documentation-intensive approach i.e. focusing much on documentation.
- But in this procedure, little attention is given on producing credible *increments* of the desired products.
- It follows the Big bang approach i.e. all FDs are delivered at once. That means, all Design Specifications are made 'OK' at once.
- Milestones are decided and passed over to all stakeholders via review meetings. These meetings may be technical or managerial.
- Lot of energy is spent on producing paper documentation to show progress versus efforts and to address the real risk issues and also the integration issues. These issues can be such as;
  - Stakeholders did not go through design.
  - Very low value to the stakeholders' opinions in the meetings but have to pay high costs on travel and accommodations.
- Many issues could have been avoided during early life-cycle phases rather that caused serious problems in later life cycle phases.

Typical Software product design Reviews :

1. Big briefing to a diverse audience

- Only very small percentage of audience understands the software programming and development.
- Briefings and documents represent few of the important assets and risks of that particular complex software.

2. A compliant design
- But, there is no tangible proof of compliance
- Compliance with unclear requirements is of little value.
3. Requirement Coverage
- Only few are the real design drivers, but many are presented as to be real
- Dealing with all requirements diverts the focus on critical drivers.
4. A design is considered 'innocent until proven guilty'

The design errors are not detected in the early phases are exposed in the later life cycle.

1.6 Conventional Software Management Performance

Most of the conventional software development generally describe the fundamental economic relationships that are derived from years of practice.

Basic Software Economics :

1. Detecting and fixing the software bugs after the delivery to customer costs 100 times more than finding and fixing the problem in early design phases.

2. You can compress i.e. minimize the software development schedules up to 25% but no more.
- Addition of staff requires more management overhead and training of the new staff.
- Some compression is sometimes proved to be troublesome to add new people.

3. For every penny, you spend on software development, you will spend the double on maintenance.
- Successful products have much higher ratios of "maintenance to development".
- A good development organization will most likely NOT spend this kind of extra money on maintenance .

4. Software development and maintenance costs are primarily the results of number of source lines of code.
- Component-based development weakens this by increasing the reuse but not in common use in the past.

5. Variations among stakeholders and especially variations in staff results in the biggest differences of software productivity.
- Development organization must always try to hire good people.
- Build the 'team concept.' With no "I" in 'team", and there must be an implicit "we."

6. Overall ratio of software to hardware costs is growing.

- Impacting these figures is the ever-increasing demand for functionality and attending the complexity.

7. Only about 15% of software development efforts are devoted to actual coding.
- And, this 15% is only for the programming. And near about, some 65% - 70% of the overall total life cycle expenses are based on maintenance.

8. Software systems and products generally cost 3 times as much per SLOC (Source Lines of Code) as individual software programs.
- Software-system products i.e. system of systems costs nine times as much.
- The more software you build, the more expensive it is per source line.

9. Walkthroughs detect 60% of the errors.
- Walkthroughs are good for catching errors, but they require deep analysis to catch significant shortcomings.
- Major problems such as performance and resource contention are also not caught.

10. 80% of the contribution is from 20% of the contributors i.e. 80/20 rule applies to many things.

Review Questions

Q.1What is software project management?

Q. 2 What was the historical perspective of conventional software development?

Q. 3 Suggest the changes in earlier and 'new' version of software development.

Q. 4 What are the problems faced due to late risk resolution?

Q. 5 Write a note on typical software development reviews?

Q. 6 Write in brief about conventional software management performance.

Q. 7 What is Late design breakage?

Q. 8 What do you mean by software economics?

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

2. Software Economics

Syllabus:

Evolution of Software Economics: Software Economics, Pragmatic Software Cost Estimation.

Improving Software Economics: Reducing Software Product size, improving software process, improving team effectiveness, improving automation, achieving required quality, peer inspections

2.1 Software Cost Estimation

- Planning and estimating are iterative processes which continue throughout the course of a project.
- Software costs often dominate computer system costs. The costs of software on a PC are often greater than the hardware cost.
- Software costs more to maintain than it does to develop. For systems with a long life, maintenance costs may be several times development costs.
- Software engineering is concerned with cost-effective software development.

Costs of Software Engineering:

- The total cost required in developing a software product can be categorized as below;

i. 60% of development costs
ii. 40% are testing costs.

- But, these costs differ depending on the type of the project being developed and the required system attributes such as the performance, reliability, flexibility, security andso on.
- The required cost in various phases also depends on the development model (either prototyping or spiral or waterfall or any other model) that is used.

Cost Estimation :

- Is the art of approximating the probable cost of something based on information available at the time.
- Leads to a better understanding of the problem.
- Improves management insight into resource allocation problems.
- Provides an objective baseline to measure progress.
- The reliability of cost estimates varies over time. The closer you get to the actual completion of a project, the estimate becomes more accurate.

Cost estimating problems occur most often because of the:

- Inability to accurately size a software project,
- Inability to accurately specify a software development and support environment,
- Improper assessment of staffing levels and skills, and
- Lack of well-defined requirements for the specific software activity being estimated

Four types of cost estimates represent various levels of reliability:

- **Conceptual Estimate :** Often inaccurate because there are too many unknowns.
- **Preliminary Estimate :** Used to develop initial budget, more precise.
- **Detailed Estimate :** Serves as a basis for daily project control.
- **Definitive Estimate :**Accuracy should be within 10% of final cost.

Cost Estimations are constructed based on the following tasks :

- Identifying the purpose and scope of the new system-New software development, software reuse, COTS integration, etc.
- Choosing an estimate type - Conceptual, preliminary, detailed, or definitive type estimates.
- Identifying system performance and/or technical goals.
- Laying out a program schedule.
- Collecting, evaluating, and verifying data.
- Choosing, applying, cross-checking estimating methods to develop the cost estimate.
- Performing risk and sensitivity analysis
- Providing full documentation.

2.1.1 Software Cost Estimation Process

Software Cost Estimation process comprises of 4 main steps :

Step 1 : Estimate the size of the development product

Size of the software may depend upon: lines of code, inputs, outputs, functions, transactions, features of the module and etc.

Step 2 : Estimate the effort in person-hours

The effort of various Project tasks expressed in person-hours is influenced by various factors such as:

- Experience/Capability of the Team members
- Technical resources
- Familiarity with the Development Tools and Technology Platform

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

Step 3 : Estimate the schedule in calendar months.

The Project Planners work closely with the Technical Leads, Project Manager and other stakeholders and create a Project schedule. Tight Schedules may impact the Cost needed to develop the Application.

Step 4 : Estimate the project cost in dollars (or other currency).

Based on the above information the project effort is expressed in dollars or any other currency.

2.1.2 Cost Estimation Techniques

1. **Expert Opinion :**

Also called as Delphi method, proposed by Dr. Barry Boehm is useful in assessing differences between past projects and new ones for which no historical precedent exists.

Advantages :

- Little or no historical data needed.
- Suitable for new or unique projects.

Disadvantages :

- Very subjective.
- Experts may do partiality
- Qualification of experts may be questioned.

2. Analogy:

Estimates costs by comparing proposed programs with similar, previously completed programs for which historical data is available.

- Actual costs of similar existing system are adjusted for complexity, technical, or physical differences to derive new cost estimates
- Analogies are used early in a program cycle when there is insufficient actual cost data to use as a detailed approach
- Compares similarities and differences
- Good choice for a new system that is derived from an existing subsystem.

Advantages :

- Inexpensive
- Easily changed
- Based on actual experience (of the analogous system)

Disadvantages :

- Very Subjective

- Large amount of uncertainty
- Truly similar projects must exist and can be hard to find
- Must have detailed technical knowledge of program and analogous system

3. Parametric :

Utilizes statistical techniques. Can be used prior to development.

Advantages :

- Can be excellent predictors when implemented correctly
- Once created, CERs are fast and simple to use
- Easily changed
- Useful early on in a program
- Objective

Disadvantages :

- Often lack of data on software intensive systems for statistically significant CER
- Does not provide access to subtle changes
- Top level; lower level may be not visible
- Need to be properly validated and relevant to system

4.Engineering :

Also referred to as *bottoms up* or detailed method.

- Start at the component level and estimate the effort required for each component. Add these efforts to reach a final estimate.
- Future costs for a system are predicted with a great deal of accuracy from historical costs of that system.
- Involves examining separate work segments in detail.
- Estimate is built up from the lowest level of system costs.
- Includes all components and functions.
- Can be used during development and production.

Advantages :

- Objective
- Reduced uncertainty

Disadvantages :

- Expensive

- Time Consuming
- Not useful early on
- May leave out software integration efforts

5. **Actual :**

Decides future costs on recent historical costs of same system.

- Used later in development or production.
- Costs are calibrated to actual development or production productivity for your organization

Advantages :

- Most accurate
- Most objective of the five methodologies

Disadvantages :

- Data not available early
- Time consuming
- Lab our intensive to collect all the data necessary

Choice of methodology depends upon :

- Type of system - software, hardware, etc
- Phase of program - Development, Production, Support
- Available data - Historical data points from earlier system versions or similar system or Technical parameters of system.

2.1.3 Cost Estimation Parameters

Various models (such as COCOMO, Costar etc.) are available for estimating the cost of software development.

All these cost estimating models can be represented on the basis of five basic parameters:

1. **Size :** The *size* of the proposed software product is weighed in terms of components i.e. ultimately in terms of the number of source code instructions or the number of functions required in developing the proposed product.

2. **Process :** The process includes the phases and activities carried out in each phase. So whatever process used to produce the proposed product is measured on the ability of the Target process to avoid unnecessary activities such as rework, bureaucratic delays, communications overhead and such other overhead activities which may delay the delivery of the product.

3. **Personnel** : This deals with the capabilities and experience of software engineering ***personnel*** (team members) in the field of computer science issues and the applications domain issues of the project. It also depends upon the personnel's' familiarity with the Development Tools and Technology Platform.

4. **Environment:** The ***environment*** constitutes of the tools and techniques that are required to develop efficient software and also to automate the process.

5. **Quality :** The required ***quality*** of the proposed product depends upon the features, performance, reliability, and adaptability of the software.

The above described five parameters (size, process, personnel, environment and quality) can be related with each other so as to calculate the estimated cost for the proposed software development:

Effort/Cost = (Personnel)(Environment)(Quality)(Size ^{process})

2.2 Software Economics

The most important aspects of ***software economics*** according to that represented in today's software cost models is the "relationship between ***effort*** and ***size".*** This relationship represents a ***diseconomy*** of scale.

This diseconomy of the software development is because the ***process*** exponent is greaterthan 1.0 i.e. the more software (that means, the more function units) you build, the more expensive it is per unit code.

2.2.1 Three Generations of Software Economics :

- The below figure demonstrates how the technology, tools, component development and processes evolved and enhanced through the three generations of software economy. 1. Con
- The estimated level of quality and personnel (person capabilities per hour) are , considered to be constant.
- The ordinate of the graph represents the software unit costs interpreted by an organization.

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

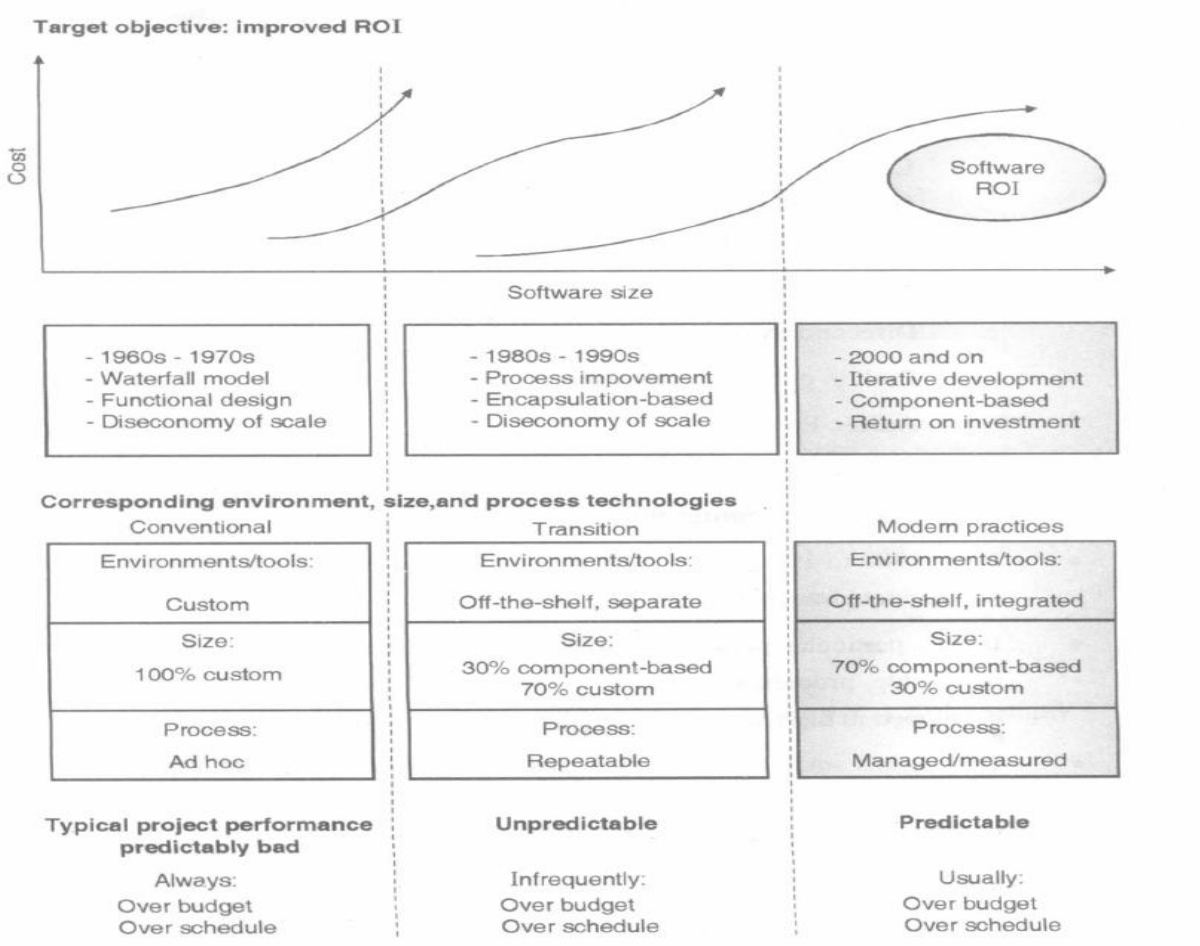


Fig. 2.2.1: Three Generations of Software Economics

1. Conventional (Craftsmanship):

- From 1960 to 1970, the conventional software development process was referred to as craftsmanship such as the Waterfall model.
- In this particular period, software development companies made use of tools, processes and components built in primitive languages.

That means, project performance was highly as expected and that too in the estimated cost and schedule.

But the quality was not as expected.

Main features :

- Custom processes and tools
- Functional design
- Custom environment (100%)

Drawbacks :

- Diseconomy of scale
- Built in primitive languages
- Ad Hoc Process
- Always over budget and over schedule

Transition (Software Engineering):

- From 1980 to 1990,the conventional software development process was called as software engineering.
- In this particular period, the software development companies used frequently repeatable processes and off-the-shelf tools, and lot of custom components developed in high level programming languages such as Java and .Net.
- That means some of the components also included commercial products such as Networking and graphical user interfaces, operating systems like Windows and Linux, database systems like Oracle and SQL Server.

- **Main features:**
  - maturity and creativity
  - research-intensive
  - Process improvement
  - Encapsulation-based
  - Built in higher-level languages
  - Repeatable process
  - Off-the-shelf environment (70% custom and 30% component-based)

- **Drawbacks:**
  - Diseconomy of scale
  - Infrequently on budget and on schedule

3. Modern practices (software production):

- Since 2000, the software development process is referred to as software production.
- Now a days, software development companies make use of managed and measured processes, integrated environments such as automated tools and technologies like the Rational Rose tool that we use for drawing UML diagrams, more of off-shelf components and few custom components.

- **Main features :**
  - production-intensive
  - automation

- economies of scale
- Iterative development
- Component-based
- Off-the-shelf environment (70% component-based and 30% custom)
- Process - managed/measured
- Usually on budget and on schedule

Throughout each of these three software economic generations, the main focus was on the *corresponding growth of technologies.* For example, consider the process advances that cannot be used successfully without the involvement of new component technologies and the enhanced tool automation.

The development organizations are constantly achieving better economies of scale as shown in every successive generation with very large projects, long-lasting software products, and business cases comprising of multiple similar projects.

The figure below gives an overview of how the return on investment (ROI) can be achieved through continuous efforts across the life cycles of various business domains.

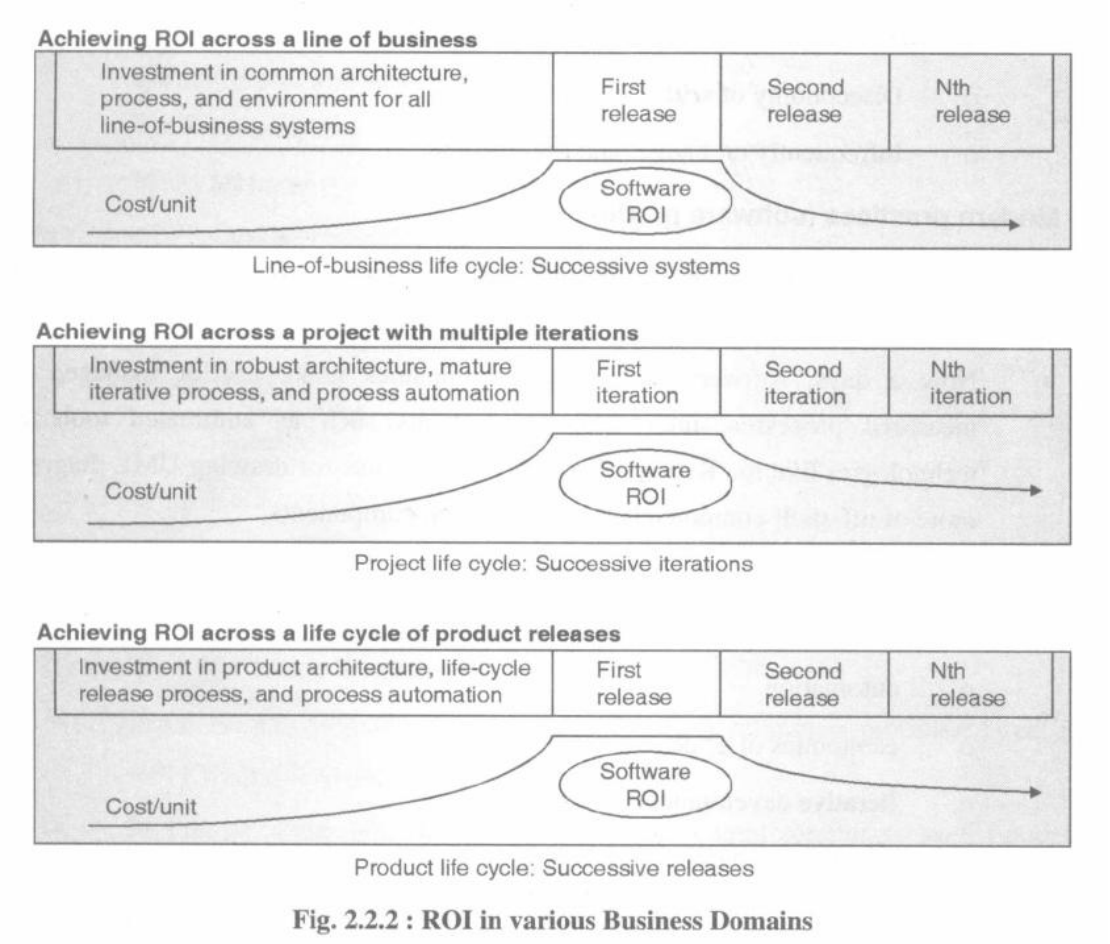


Fig. 2.2.2 : ROI in various Business Domains

2.3 Pragmatic Software Cost Estimation

- Major issue of software cost estimation is the inability of developing well-documented case studies in the projects that use iterative development approach (also called as Rational Unified Process (RUP)).
- Software development companies don't have a clear idea about software metrics and measures which means that the data storage and retrieval process is not clear in terms of data consistency and the actual quality attributes of the projects don't match with planned attributes.
- It is easy to collect related set of data within one particular organization but it is difficult to collect the related data from different companies following different development processes, different programming languages and different domains.
- Developers and customers always had an argument on software cost estimation models and tools.

Three most common topics of their argument are :

1. **Selection of Cost Estimation Model**

- There are various cost estimation models (COCOMO, COSTXPERT, CHECKPOINT, SLIM, ESTIMACS, Knowledge Plan, SEER, SOFTCOST, Costar, REVIC, Price-S, Pro QMS etc.) that are based on statistically derived cost estimating relationships (CERs) and various estimating methodologies.

2. **Whether to measure software size on the basis of lines of code (LOC) or function points?**

- Most of the cost estimation models are bottom-up i.e. substantiating a target cost rather than top-down i.e. estimating the 'should' cost.

- The below figure depicts predominant cost estimation practices where:
  - First, project manager defines the target cost of the software.
  - Later, he manipulates the parameters and does the sizing until the target cost is justified.

- The target cost is defined so as :
  - to win the proposal,
  - to solicit the customer funding,
  - to attain the internal corporate funding, or
  - to achieve some other goal.

- The process described in this figure forces the software project manager to check the risks associated in achieving the target costs and also discuss this problem with other stakeholders.

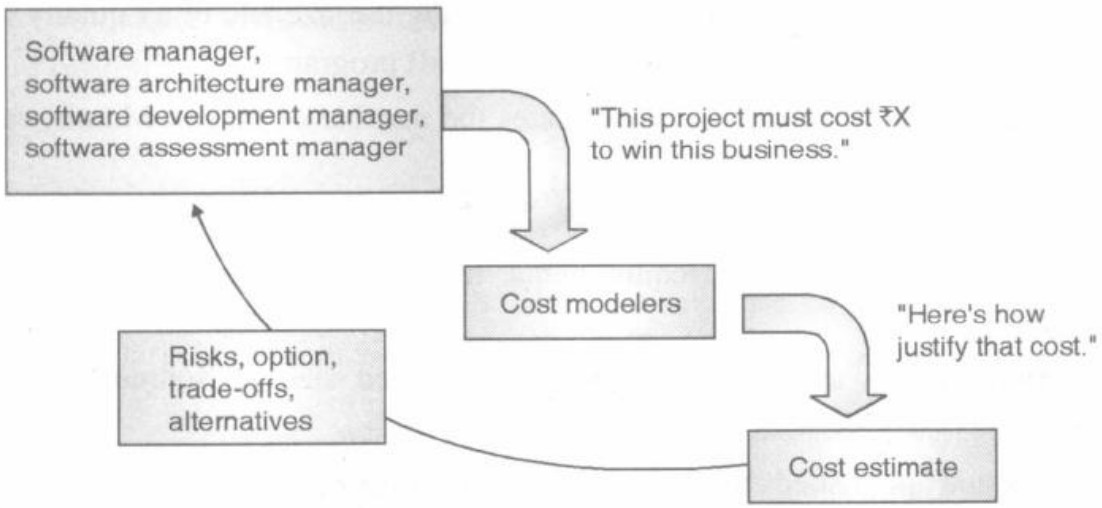


Fig. 2.3.1 : Predominant Cost Estimation Process

3. Factors that lead to good cost estimation
   Good software cost estimation may be based on the following attributes:

- The cost is estimated collectively by the project manager, architectural team, development team, and test team.
- The estimated cost is acknowledged and supported by all stakeholders.
- The cost estimation is based on some well-defined model.
- The cost estimation is based on the databases of similar type of project experiences that use similar methodologies, similar technologies, and similar environment with similar type of stakeholders.
- The key risk areas are detected and accordingly the probability of success is determined.

2.4 COCOMO Model

- COnstructive COst MOdel (COCOMO) is one of the earliest cost models widely used by the cost estimating community.
- COCOMO was originally published in Software Engineering Economics by Dr. Barry Boehm in 1981.
- COCOMO is a regression-based model that considers various historical programs software size and multipliers.
- COCOMO's most fundamental calculation is the use of the Effort Equation to estimate the number of Person-Months required in developing a project.
- COCOMO stands for COnstructive COst MOdel. It is the oldest cost estimation model that is popularly used in the process of cost estimation.
- COCOMO was first published by Dr. Barry Boehm in 1981.

- COCOMO model estimates the cost by considering the size and other quality aspects of the similar type of historical (previously developed) programs.
- COCOMO calculates Efforts i.e. it estimates the number of Person-Months required in developing a project.

Number of person months * loaded lab our rate = Estimated Cost

- Most of the other estimates (requirements, maintenance, etc) are derived from this quantity.
- COCOMO requires as input the project's estimated size in Source Lines of Code (SLOC).
- Initial version published in 1981 was COCOMO-81 and then, through various instantiations came COCOMO 2.
- COCOMO 81 was developed with the assumption that a waterfall process would be used and that all software would be developed from scratch.
- Since then, there have been many changes in software engineering practice and COCOMO 2 is designed to accommodate different approaches to software development.

The COCOMO model is based on the relationships between the two formulas:

- **Formulae 1:** Development effort is based on system size. MM = a.KDSI b
  where,

MM is the effort measured in Man per Moths

KDSI is the number of Source Instructions Delivered in a Kilo (thousands).

- **Formulae** 2: Effort (MM) and Development Time. TDEV = c.MM d
  where,

TDEV is the development time.

In both the above formulas, we have used the coefficients a, b, c and d which are dependant upon the 'mode of development'. According to ***Boehm, the mode of development can be classified into following 3 distinct modes:***

1. **Organic** mode of development - talks about the projects that involve small development teams whose team members are familiar with the project and work to achieve stable environments. This category includes the projects like the ***payroll systems.***

2. **Semi-detached** mode of development **-** talks about the projects that involve mixture of experienced team members in the project. This category includes the projects like the ***interactive banking system.***

3. **Embedded** mode of development **-** talks about the complex projects that are developed under tight constraints with innovations in it and have a high volatility of requirements. This category includes the projects like the ***nuclear reactor control systems.***

Drawbacks :

- It is difficult to accurately estimate the KDSI in early phases of the project when most effort estimates are still not decided yet.
- Easily thrown misclassification of the development mode.
- Its success largely depends on tuning the model to the needs of the organization and this is done based upon the historical data which is not always available.

Advantages :

- COCOMO is transparent that means we can se it working.
- Allows the estimator to analyse the different factors that affect the project costs.

2.4.1 C0C0M0 2 Model

COCOMO 2 constitutes of sub-models so as to produce in detail software estimates. The sub models are listed as follows;

- Application composition model. It is applied when software is being developed from existing parts.
- Early design model. It is applied when system requirements are gathered and concluded but design has not yet started,
- Reuse model. It is applied when software is being developed using the reusable components so as to compute the effort of integrating these components,
- *Post-architecture model.* It is applied when once the system architecture has been designed and when more system information is gathered. Multipliers reflect the capability of the developers, the non-functional requirements, the familiarity with the development platform, etc.

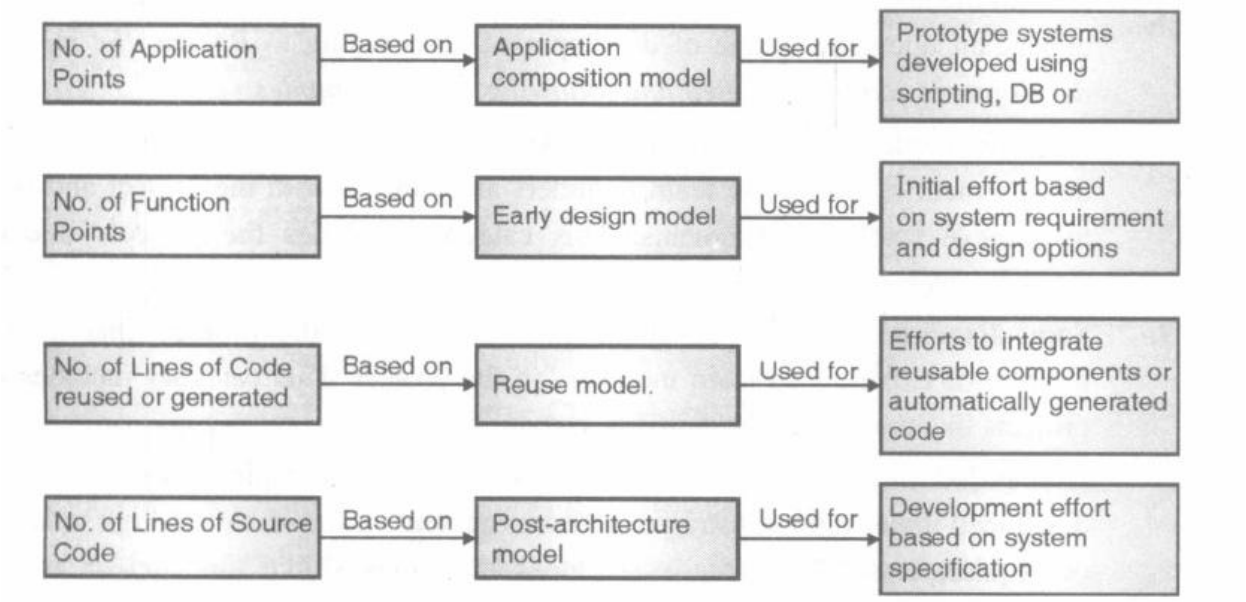


Fig. 2.4.1 : Use of COCOMO2 Model

Product attributes describe the required characteristics of the software product being developed.

Computer attributes describe the constraints imposed on the software product by the hardware platform. Personnel attributes describe the experiences and capabilities (of the project development team members) that are taken into account. Project attributes describe particular characteristics of the project. Estimating the calendar time required to complete a project and when staff will be required using a COCOMO 2 formula TDEV = 3 ' (p m)(0-33+0-2*<b-101»

- PM is the effort computation and B is the exponent (B is 1 for the early prototyping model). This computation predicts the nominal schedule for the project.
- The time required is independent of the number of people working on the project. Improving Software Economics The software economics can be improved by using a 'balanced' approach as the key. The Five Key parameters that can help in improving the software economics are;

Reducing the product size (Number of Lines of Code) and complexity of the software

1. Improving the software development process
2. Using more-skilled personnel i.e. improving the team effectiveness.
3. Creating better environment by improving the automation with more appropriate tools and technologies.
4. Achieving required quality by peer inspections

Table 2.4.1 : Trends on which the Five Key Parameters depend

Five Key parameters that effect the Cost Model	Trends
Size: describes abstraction and component based development technologies	High level programming Languages (C++, Java, VB, Object Oriented Methods and Visual modeling (analysis, design and programming) Reusability Commercial Exponents Packages
Process: involves methods and techniques	Iterative Development Process Maturity Models such as CMM Architecture-first development
Personnel: describes the effectiveness of the development team	Training to develop the personnel skills Team work Win-win culture
Environment: involves automated tools and technologies.	Integrated tools such as compiler, editors, and debuggers. Hardware performance Automated coding, documentation, testing and analyses.
Quality: describes the Performance, reliability, accuracy issues	Hardware platform performance Peer inspections Statistical quality control

2.5 Reducing Software Product size

- The more the number of lines of code, the larger becomes the size of the product; larger the size of the product, more expensive becomes the product and these expenses are measured 'per line'.
- The most proper way of improving the affordability and ROI (return on investment) is to develop a product that achieves the design goals with minimum number of LOC (Lines of Code) i.e. minimum amount of source code.

Parameters that affect the size complications are:

- Use of Component-based development i.e. breaking up the software product into simple modules and these modules are coded separately and after complete coding is over, all these modules are then integrated together. This decomposition of the product first and then later integration raises complications in the code.
- Automatic code generation - Various design tools generate automatic code along with the modeling of the designs. Such tools sometimes generate lot of unwanted code.
- Graphical User Interface (GUI) builders include the code that is needed to build an easy to access user interface. In such process, more lines of code are included in the program.

- 4th Generation Languages (4GLs) make use of classes, structures, dynamic memory allocations and many such new concepts that increase the complexity of the code.
- Object Oriented (OO) Modeling Languages used for analysis, design and modeling also increases the complexity of the software.

2.5.1 Reducing Software Product size - Languages

UFP (Universal Function Points):

- These points are language independent.
- The basic units of the UFP are external user inputs, external outputs, internal logical data groups, external data interfaces, and external inquiries. SLOC (Source Lines o f Code) m etrics :
- These metrics are useful estimators when a solution is formulated and programming language is known.

Table 2.5.1 : Various Comparisons of Function points to lines of code based upon the programming language used.

Language	SLOC per UFP
Assembly	320
C	128

Language	SLOC per UFP
Fortran 77	105
Cobol 85	91
Ada 83	71
C++	56
Ada 95	55
Java	55
Visual Basic	35

Advantages :

- Use of higher level programming languages reduces the size, thus, the 'level of abstraction' also changes allowing more focus on architecture.
- The reduced size makes it easier to understand, reuse, maintain and import the packages of classes and objects.
- But, these higher-level abstractions often use high storages and communication bandwidths.

2.5.2 Reducing Software Product size - OO Methods and Visual Modelling

- OO (Object Oriented) technology reduces the size of the program. How the use of OO methodology helps in reducing the size can be summarized as below;

Benefits of using OO methodology :

- An OO methodology improves the team work and interpersonal communications by improving the understanding between the end users and developers of the system. This ultimately increases the productivity and quality.
- OOSE (OO software Engineering) is achieved using the OO modeling languages like UML and the configurable processes like RUP (Rational Unified Process) which involves iterative development.
- OO methodology supports continuous integration of subsystems, classes, interfaces thus, increasing the chances of early detection of risks and incremental corrections without hampering the stability of the development process.

- OO methodology allows 'Architecture first approach' in which integration is an early and continuous life-cycle activity that brings stability in development, allows development and configuration of components in parallel.

- OO architecture creates a clear separation between the unrelated elements of a system, and includes firewalls that prevent a change in one part of the system that may be caused due to the errors in other part of the system thus, rending the structure of the entire architecture.

2.5.3 Reducing Software Product size - Reusability

- The 'reusability' is mostly implemented with stored functions and subprograms. There are many forms of reuse:
- Reuse of *Old stuff* such as data descriptions, document, and a collection of similar-old artifacts, designs, and architectures.
- Reuse of *Functions*, *classes* from any phases of software development
- Reuse of *Common architectures* such as common processes, and common environments. Such reuse is very common during huge project development.
- Differences in OS platforms and HAV environments such as the middleware, and also the GUI Builders have hampered the reuse potentiality.

Examples : Common Microsoft platforms, Linux MACs resulting from distributed applications.

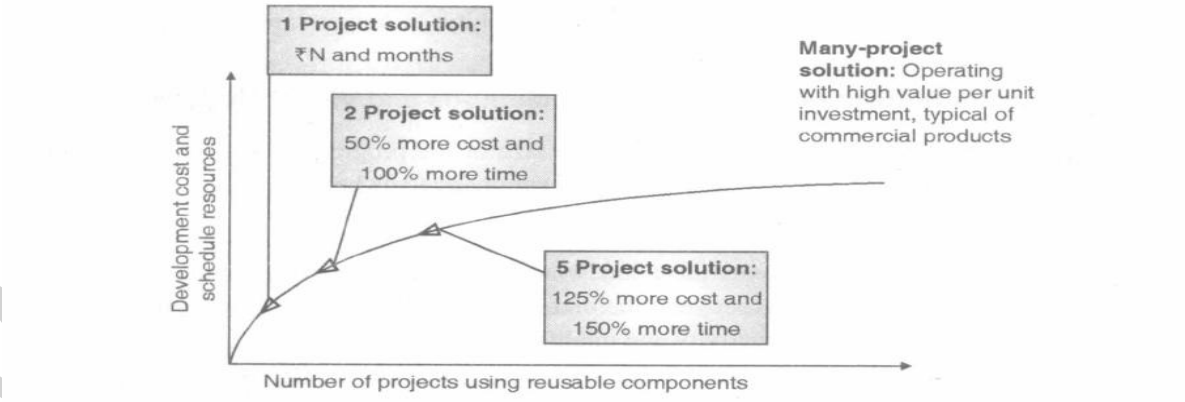


Fig. 2.5.1 : Graph showing the Development cost in the projects using Reusable Components

- The main reason for Reuse is 'lack' of money.
- Costs needed to build reusable and configure reusable components.
- Reuse is implemented across many projects if they are similar.
- Few commercial organizations sell commercial components encouraging the Reusability.

2.5.4 Reducing Software Product size - Commercial Components

- Main advantage of using Commercial components is that it saves custom development efforts
- Commercial components usually need tailoring i.e. they need finishing before they are used.
- Commercial components have an impact on quality, cost, supportability, and the architecture.

Table 2.5.2 : Pros and cons of Commercial Components v/s Custom Software

Approach	Pros (Advantages)	Cons (Disadvantages)
Commercial Components	Predictable costs largely used and is a matured technology Available now a days supports organizations Hardware and Software independence Rich in functionalities	Needs frequent upgrades and maintenance Charges Up-front license fees Charges Recurring maintenance fees Highly Depends on vendor Run-time efficiency is little bit less Lot of functionality constraints Integration is always needed at each step Inclusion of Unnecessary features that consume extra resources mostly, inadequate reliability and stability Multiple vendor incompatibility
Custom Development	Freedom of making changes Provides smaller and simpler implementations Provides better performance Control on development and enhancement	Unpredictable development Cost Unpredictable availability date Undefined maintenance model Often immature and easily breakable Single-platform dependency Consumes expert resources

2.5.5 Reducing Software Product size - Packages

- Various elements (use cases; classes, other model elements) of analysis model are categorized into packages that are given a representative name.

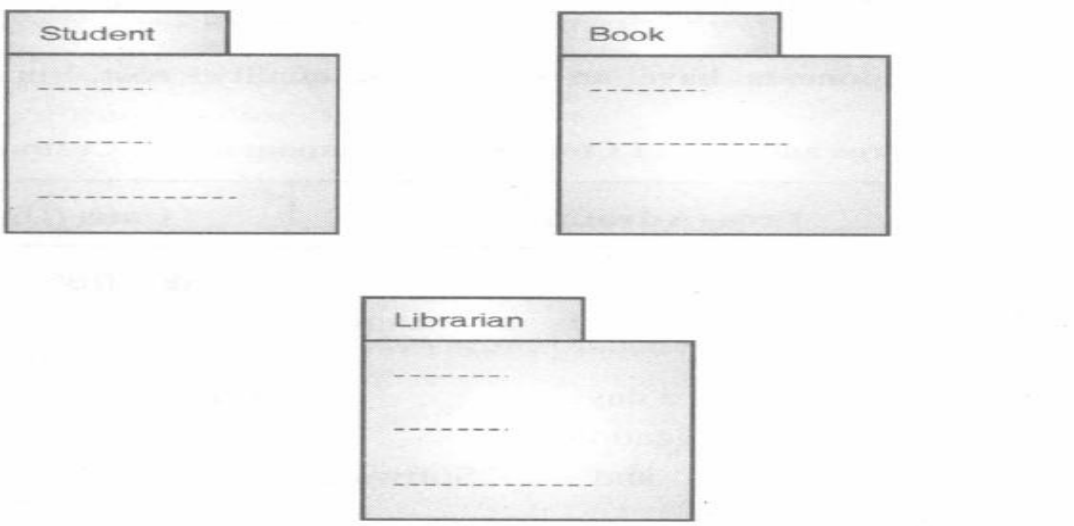
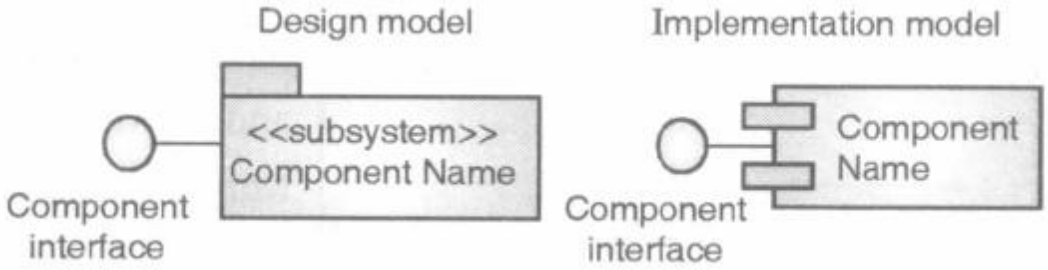


Fig. 2.5.2 : Packages

Example : Student class, Book class, Librarian class.

- Package is a model element that can contain other model elements - use case models, classes, objects, subsystems, components, other model elements and packages.
- The Components help to model the physical aspect of an Object-Oriented software system representing the relationship between various components.



• A component diagram contains components and dependencies. The dependencies between the components show how changes made to one component may affect the other components in the system. Dependencies in a component diagram are represented by a dashed line between two or more components.

The component diagram contains:

- *Components* are denoted by rectangle with two smaller rectangles protruding from its left side. They represent the subsystems in the design model.

- *Dependencies* are denoted by dashed lines between two or more components. They represent how changes made to one component may affect other components of the system.

Example : Component Diagram for Library Management System

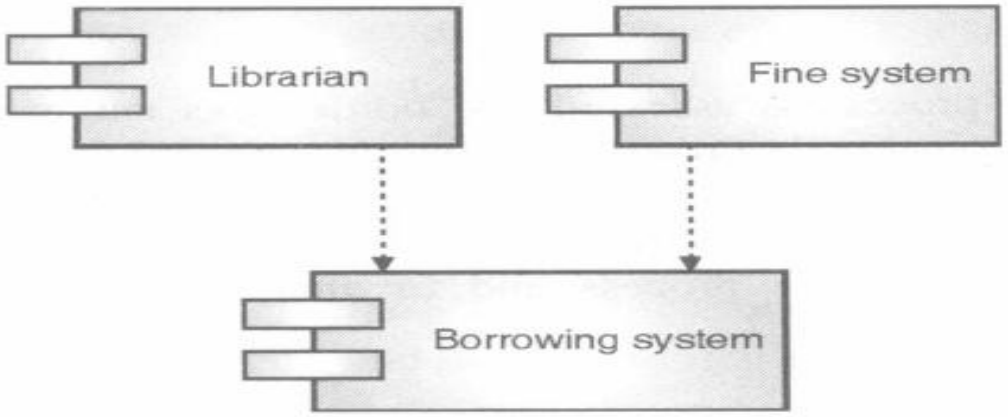


Fig. 2.5.3 : Component Diagram for LMS

The above diagram shows 3 main components of the library management system in which the first component (Librarian) manages each student's profile and also who manages library items (issuing and returning). Second component (Fine system) manages fines applied to the student who exceed the borrowing period. Third component (Borrowing system) manages all borrowing items.

2.6 Improving Software Process

• It involves understanding the existing processes and introducing changes in it so as to improve the product quality, reduce costs and accelerate schedules.

• Process improvement work focuses mostly on defect reduction and improving the development process.

• Process improvement is a cyclic activity as shown in the below figure :

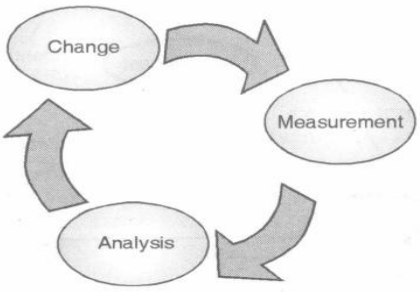


Fig. 2.6.1 : Process Improvement Cycle

It involves three principal stages :

1. Process measurement:
Attributes of the current process and product are measured. These form a baseline for assessing the improvements.

2. Process analysis :

- The current process is assessed and bottlenecks and weaknesses are identified. Process models that describe the system process are usually developed during this stage.
- It is about studying the existing processes to understand the relationships between different parts of the process and to analyze i.e. compare them with other processes.

3. Process change :

Changes to the process that have been identified during the analysis are introduced.

2.6.1 Need of Process Improvement:

- SPI framework defines the characteristics of an effective software process in an effective manner.
- SPI framework is used to assess existing organizational approach of software development against those characteristics.
- SPI framework defines a meaningful strategy for improvement.
- SPI framework transforms the existing approach of software development into more focused, more repeatable and more reliable process.
- SPI framework assesses the *maturity* of an organization's software development process and identifies its maturity level.

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

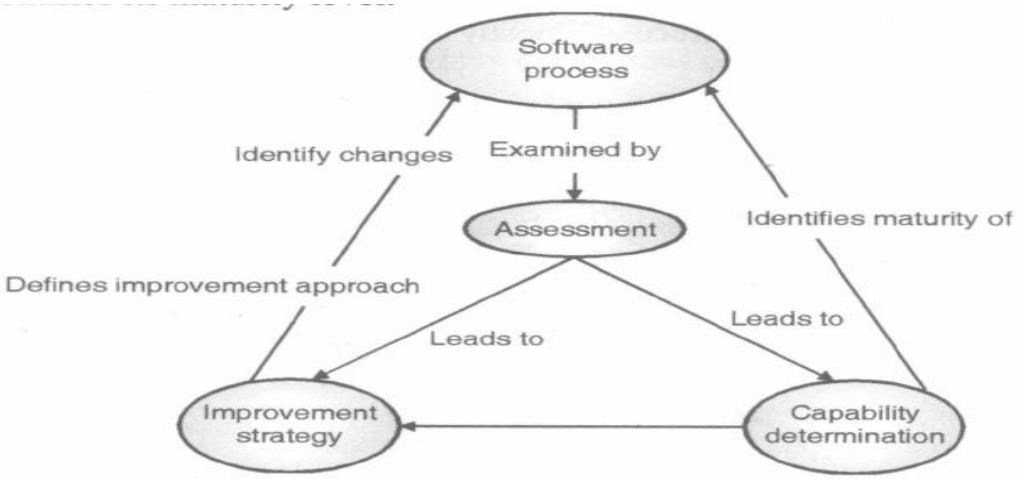


Fig. 2.6.2 : Key Elements of SPI Framework and their Interrelationship

2.6.2 Objectives of Software Process Improvement (SPI):

- To enhance concurrency activities - Few activities of a project are done in *parallel* others in *sequential.* But if the parallel activities are carried out, this will reduce the cost and schedule.
- To minimize the *overhead* and instead use these efforts towards *production* activities.
- Few activities are *overhead* and others are *production* activities
- The *Overhead activities* include project planning, progress monitoring, risk management, configuration and version control, quality control, component integration, testing, rework, personnel training, etc.
- The *Production activities* include the project itself that composes of requirements elicitation, modeling, analysis, and design and implementation activities.
- To eliminate the *late* rework and fixing.

Advantages :

A good-quality process reduces:

The required efforts and ultimately reduces the required schedule

The required Project time schedule but, yet with improved quality.

The three main improvements are:

- Improves efficiency of each step in the process
- Eliminates some of the overhead steps in the process which yields improved ROI.
- Carries out same number of steps but makes use of concurrency wherever possible

Table 2.6.1: Three levels of Processes and their Attributes

Attributes	Meta-process	Macro-process	Micro-process
Subject	Line of business	Project	Iterations
Objectives	Line-of-business Profitability Competitiveness	profitability Risk management Project budget and schedule Project quality	Resource management Risk resolution Milestone budget, schedule quality
Audience	Acquisition of authorities, customers Organizational management	Software project managers Software engineers	Subproject managers Software engineers
Metrics	Project predictability Revenue, market share	Predictability of budget and schedule Major milestone success Project scrap and rework	Predictability of budget and schedule Major milestone progress Release and iteration scrap and rework
Concerns	Bureaucracy v/s standardization	Quality v/s financial performance	Content v/s Schedule
Time scales	6 to 12 months	1 to many years	1 to 6 months

2.6.3 CMMI Process Improvement Framework

- Capability Maturity Model (CMM) is a maturity model applied within the context of SP I framework.
- CMM is a specific approach taken for quality assurance.
- The CMMI (Capability Maturity Model Integration) is used to implement Quality Assurance in an organization.
- The CMMI describes an evolutionary improvement path from an adhoc, immature process to a mature, disciplined process which describe the key elements of an effective software process.

- The CMMI includes key practices for planning, engineering, and managing th software development and maintenance which helps in improving the ability of organizations to meet goals for the cost, schedule, functionality, and product quality.
- The CMMI helps in judging the maturity of an organization's software development process and compares it to the state of practice of the industry.
- The CMMI categorizes five levels of process maturity :

Level 1: Initial : An adhoc software process is used where only few of the processes are defined and where the project accomplishment success depends upon individual team member's efforts.

Level 2: Repeatable : A project management process is derived to track the utilized cost, schedule, and efforts. This same process is used again and again to develop the projects of similar applications so as to repeat the success achieved in the earlier projects.

Level 3: Defined : Both the activities i.e. project management and software engineering are well documented and integrated.

Level 4: Managed : It includes better understanding and planning of software process and product quality.

Level 5: Optimize : The developing organization's focus is on continuous software process improvement (SPI) which is achieved by continuous process feedback and by incorporating innovative ideas and technologies in project development.

- At this level, the entire organization focuses on continuous Quantitative feedback from previous projects which is used to improve the project management.
- The software process at this level can be characterized as continuously improving the process performance of their projects.
- Improvement is done both by incremental enhancements in the existing process and by innovations using new technologies and methods.

These levels are decomposed into several key process areas.

- **Process Change Management** : To identify the causes of defects and prevent them from re-occurring.

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

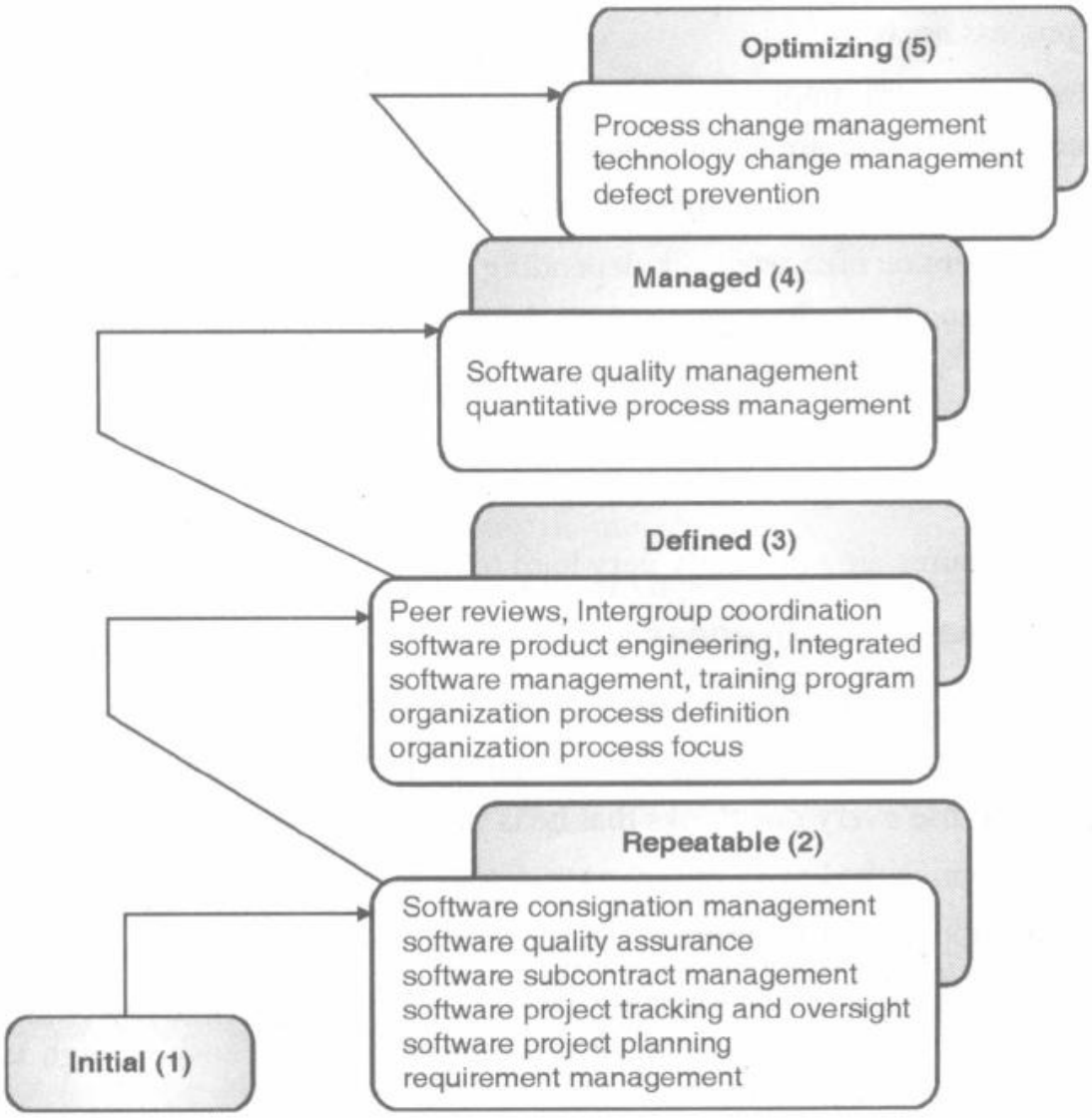


Fig. 2.6.3 : CMMI and Key Process Areas

- **Technology Change Management** : To identify beneficial new technologies and incorporate them in an orderly manner.
- **Defect Prevention :** To continuously improve the process in order to improve the quality productivity and thus decrease development schedule and cost.

The Process Area Activities performed include :

A software process improvement program is established which motivates the members of the organization to improve the processes of the organization. The group responsible for the organization's software process activities controls the software process improvement activities also. The organization develops and maintains the plan for improving the software process according to the documented

procedure. The software process improvement activities are performed corresponding to the software process improvement plan. Members of the organization participate in teams to improve software process for assigned process areas. The software process improvements are installed to determine their benefits and effectiveness before they are introduced into normal practice. Records of software process improvement activities are maintained. Software engineers receive feedback depending on the status and results of the software process improvement activities on an event-driven basis.

Drawbacks of CMM :

The *CMM* does not describe how to create an effective software development organization. The traits it measures are practically very hard to implement in an organization.

2.6.4 Improving Team Effectiveness

It has been observed that poor personnel yields poor productivity. But on the contrary, it is also impossible to manage a team with all stars in it. There are always disputes in such a great team because everyone thinks that he is more intelligent than the other. Managing the team is the key to improve the team effectiveness. Best pragmatic approaches to improve the team effectiveness are:

- **Balance :** A project development team must constitute highly talented people in key positions and less talented in other positions. It is easy enough to manage such a balanced team.
- **Coverage :** A project development team constitutes of strong skill people in key positions. Boehm's Principles (Recommendations) in order to improve the team 's effectiveness:
  - **Principle of top talent** : use talented and less number of people i.e. 'use better and fewer people'.
  - **Principle o f job matching (skills and motivations)** : Individuals in the development team must have a vision of promotion right from the programmer to project manager or to architect or to designer. All individuals in a team don't have same skill sets - Great programmers are not necessarily great managers and conversely,
  - **Principles of Career Progression :**

i. An organization does best in the long run by helping its staff to self actualize.
ii. Organization training greatly contributes in improving the productivity.
iii. Posting for new jobs must depend upon their skills and previous work area.
iv. Organization should focus on the factors that are the prime motivators such as increments, bonus and etc.

  - **Principle of team balance:**

i. Select people who will complement and go with with one another.
ii. It represents the balance of : raw skills (intelligence, objectivity, creativity, analytical thinking)
iii. Psychological makeup (leaders and followers; risk takers, visionaries and nitpickers)

  - **Principle of Phase-out:**

- Disrupt team balance, horribly de-motivating.
- Availing a nonconformist in the team doesn't benefit anyone.

Overall team guidance:

- A *culture o f teamwork* is necessary where people complement one another and go with each other.
- *Balanced* Teamwork
- Strong and 'knowledgeable' *leader* (Project Manager) is essential:

Required Project Manager Skills:

- ***Hiring skills.*** Selecting right person for the right job.
- ***Customer-interface skill.*** Avoiding adversarial relationships among stake-holders is must for success.
- ***Decision-making skill.*** Consider diverse opinions and make no partiality

- Team-building skill.
- Keep the team together.
- Recognize individual needs and excellent performers
- Nurture the new comers
- Facilitate contributions from everyone and make every individual feel that he is important.

- ***Selling skill.*** A successful project manager must:
  - Sell the stakeholders based on decisions and priorities,
  - sell candidates on job positions,
  - sell changes to the status quo in the face of resistance, and
  - sell achievements against objectives.
  - Practically, selling requires continuous negotiation, compromises, and patience.

Problems faced while achieving Team Effectiveness :

- It may not be possible to appoint the ideal people to work on a project because:
  - Project budget may not allow for the use of highly-paid staff;
  - Staff with the appropriate experience may not be available;
  - An organisation may wish to develop employee skills on a software project.
- Managers have to work within constraints especially when there are shortages of trained staff.
- The number o f people working on a project varies depending on the phase of the project.
- As more people work on the project, the more total effort is required.
- Number of staff required can't be estimated by dividing the development time by the required schedule.

2.7 Improving Automation

- The environment that is used means the tools and technologies that are used have a drastic impact on productivity and effort and thus, ultimately has an impact on schedule and cost also.

- Large numbers of tools are available in the markets that are required for supporting a process. But...
- Make careful selection of the right combination of tools and
- Recognize the tools that are important for process automation.
- Highly integrated tools facilitate proper management of the process
- A prime motivation for the staff is to train them to work with the modem tools and learn about the environment.
- Yields robust and integrated development process
- Hires talented and right people and equips them with modern tools.

2.7.1 Problems Faced while Improving Automation :

- While buying tools, be careful of tool vendor claims.
- Prior to using the tools, they must be integrated into the development environment.

The table below represents the General Quality improvements realizable with a modern process.

Table 2.7.1 : General Quality Improvements with a Modern Process

Quality Driver	Conventional Process	Modern Iterative Processes
Requirements misunderstanding	Misunderstandings are discovered late	Misunderstandings are discovered and resolved early
Development risk	Risks are Unknown until late	Risks are detected and resolved early
Commercial components	Mostly unavailable	Available but the tradeoffs must be resolved early in the life cycle
Change management	Late in life cycle; chaotic and nasty	Early in life cycle; straight-forward and benign
Design errors	Such errors are discovered late	Any type of errors are resolved early
Automation	Mostly error-prone manual procedures	Mostly automated and error-free artifacts
Resource adequacy	Unpredictable	Predictable
Schedule	Unpredictable	Tunable to quality, performance, and technology
Target performance	Paper-based analysis or separate simulation	Executing prototypes, early feedbacks, quantitative understanding
Software process rigor	Document-based	managed, measured, and tool-supported

- Key practices that improve overall software quality:
- Focus on requirements driving the process - it addresses the critical use cases early in the life cycle and traceability late in the life cycle, focuses on requirement completeness.
- Use metrics and indicators to measure the progress and quality of the architecture as it evolves from a high-level prototype to a fully compliant product
- Use integrated life-cycle environment that facilitates early and continuous configuration and change control, fast design methods, document automation, and regression test automation.
- Use visual modeling and Higher Level Language (HLL) that supports architectural control, abstraction, design reuse, reliable programming, and self-documentation.
- Early and Continuous insight into performance issues through demonstration-based evaluations.

Performance issues :

- Be careful with commercial and custom-built components
- Performance analysis can degrade as we progress through our process.

2.8 Peer Inspections

- Conducting peer Inspections is a very old way of verifying the results.
- It is suitable and good in cases where there is a need to nurture less-experienced team members.
- It is also useful in catching the real bad errors eariy in the life cycle phases.
- Inspections can be applied at various phases during a development life cycle :

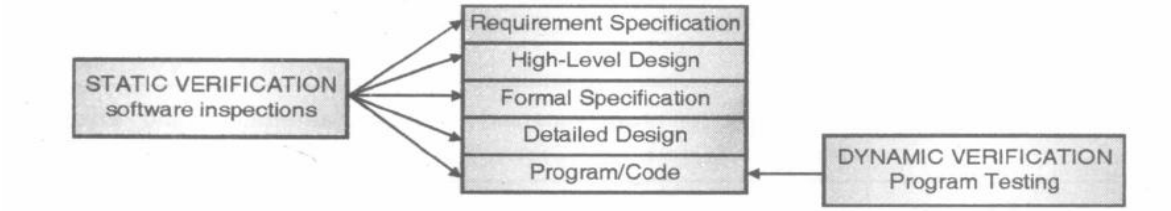


Fig 2.8.1: Software Inspection applied at any phase v/s Program Testing applied only at coding phase

- Inspections for *transitioning* the *engineering info* from one artifact set to another so as to assess the consistency, feasibility, understandability, and technology constraints involved in the engineering artifacts.

Example:

1. Analysis classes will transit into design classes which will then become the part of packages or components. In doing so, if any desired functionalities are lost then they are traced and re-included.
2. Inspections for demonstrating major milestones. This forces the artifact assessment against tangible criteria for relevant use cases.

3. Inspections of the Environment tools
4. Life-cycle testing provides insight into requirements compliance.
5. Inspections manage the changes and studies about how change requests can impact both quality and progress goals.

Inspection pre-conditions:

- A precise specification must be available.
- Team members must be familiar with the organisation standards.
- Syntactically correct code or other system representations must be available.
- An error checklist should be prepared.
- Management must accept that inspection will increase costs early in the software process.
- Management should not use inspections for staff appraisal i.e. finding out who makes mistakes.

Inspection Process:

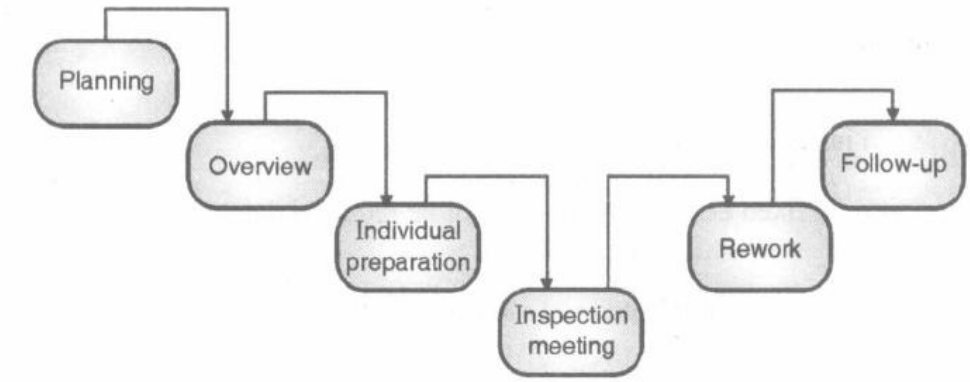


Fig. 2.8.2 : Inspection process

Step 1 : Planning :

- Selection of the review team which must include not more than five people.
- Decide who will be the moderator and what his responsibility.
- Preparing the package (work product and supporting documents) that is to be distributed among the team members for their self study.

Step 2 : Overview :

- All team members individually review the work product
  - They answer the checklist according to the given guidelines and
  - They note down the issues that they find in a self-preparation log

Step 3 : Preparation :

- Each reviewer studies the project individually.
- He notes down the issues that he has come across while studying the project.
- He decides how to put up these issues and makes a note of it.

Step 4 : Inspection Meeting :

- The reviewer reads line by line of the product
- Participants can raise any type of issue at any line
- Discussions are done to identify if any defect
- Scribe records all the decisions made in the meeting
- Scribe gives the list of detected defects (and issues) to the developer
- If there are only few defects, then the work product is accepted b; else it is sent for making changes in it and to undergo another review.
- The meeting does not propose any solutions, but if there are nay suggestions, then they are recorded and given to the developer.
- A summarized report of the meeting (in our language we say it as MOM i.e. minutes of meeting) is prepared which helps in effective evaluation of the product.

Step 5 & 6 : R ework and Follow Up :

- Author fixes the detected defects by making modifications in it.
- Once the errors are fixed, author gets it OKed by the moderator.
- Re-inspection on fixed errors may or may not be required depending upon the criticality of the problem ..
- Once the rework is completed and addressed satisfactorily, the collected data is submitted.

Inspection Roles :

Author (Owner) of the Code:	The programmer or designer responsible for producing the program or document. Responsible for fixing defects discovered during the inspection process.
Inspector	Finds errors, omissions and inconsistencies in programs and documents. May also identify broader issues that are outside the scope of the inspection team.
Reader	Presents the code or document at an inspection meeting.
Scribe	Records the results of the inspection meeting.

Chairman or Moderator	Manages the process and facilitates the inspection. Reports process results to the Chief moderator.
Chief Moderator	Responsible for inspection process improvements, checklist updating, standards development etc.

Advantages:

The goal of this method is to detect ail faults, violations, and other side-effects. This method finds out more and more number of defects by:

- A complete preparation (studying the documents) is done by authors and other reviewers before conducting inspection.
- As a group of people are involved in the inspection procedure, multiple diverse views are enlisted.
- Every person of the inspection team is assigned a specific role.
- The reader in the inspection reads out the document sequentially in a structured manner so that all the points and all the code is inspected thoroughly.

Disadvantages :

- Needs lot of time as it first involves some special time for preparations also prior to conducting formal meetings.
- Logistics and Scheduling always is a point for issue since these tasks constitute of lot of people.
- Checking every line of code is not possible every time even though it ensures the correctness of the logic, avoids the side-effects and appropriately handles the errors.

Inspection Issues:

- Ensure that complex and critical components are really inspected and verified by the primary stakeholders.
- It is difficult to really look at all artifacts.
- Inspecting too many artifacts will increase the cost.
- Mostly, many of the artifacts don’t deserve any scrutiny.
- Most inspections end up looking at style and simple semantic issues rather than inspecting real issues.

Usually, most of the quality factors and other important features of the project such as system, performance, concurrency, distribution, etc. can be discovered through following activities:

- Analysis and prototyping.
- Constructing design models will help in tracing the missing requirements and architectural constraints.

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

- Transitioning the current state of the designs into an executable implementation
- Illustrating the current implementation strengths and weaknesses in context of critical subsets of use cases and scenarios.
- Incorporating the user feedbacks into the models, use cases, implementations, and plans.

Review Questions

Q. 1 What is software economics and list out different cost estimation models.

Q. 2 Describe the cost estimation process.

Q. 3 Write short notes on cost estimation techniques.

Q. 4 List and describe the cost estimation parameters.

Q. 5 What is cost estimation and explain C O CO M O model.

Q. 6 Explain the three generations of software economics.

Q. 7 What are the various way of reducing the software product size. Explain in brief.

Q. 8 What is the need of improving the software process.

Q. 9 What are the benefits of improving team effectiveness and how to improve it.

Q. 10 How to achieve required quality?

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

UNIT II

3.The Old Way and the New

Syllabus :

The principles of conventional software engineering,

Principles of modern software management,

Transitioning to an iterative process

3.1 Introduction

As we know, every product in the market grows and gets updated every day. Each and every part of the product renews and re-polishes every minute.

Most of the organizations generate the new version of the older product by making various additional advanced changes in it. If we consider the example of the mobile company, one mobile comes in this one month, and the next generated advanced mobile comes soon in the next month. Following is the example of the Samsung mobile company which launched 'Wave I' in 2010 which was later upgraded with additional varieties (such as enlarged size) and came up in the market as 'Wave II'.

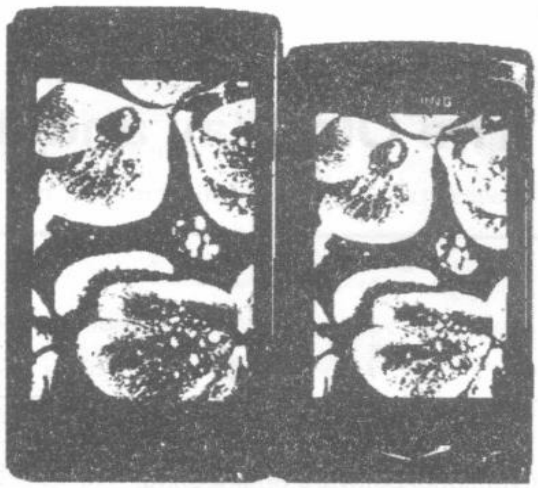


Fig. 3.1.1 : Upgradation Versions: Samsung's Wave II and Wave I mobile

Many years ago, in case of software companies, they can make naming to their upgraded version with 'x' suffix to it, even they cannot know the meaning of the 'x'. For example: lx, 2x and so on.

There are many parameters, by which software industry can improve the software economical development;

1. Require and Meet:
Initially, whatever matter was available with the developers, only by using that, the software product was generated. But now as per customer requirements and for his easy handling, the software market meets his requirements by meeting him.

2. Easy handling

It is seen that today's customer can easily handle the overall software product by which earlier traditional workloads on him became very less. **Automation** came in picture by which customer can take more relaxation by putting his workload on the automatic planners.

3) Team Building
Many departments can handle different units of each different phase. So software project is able to achieve its goals by using different 'thinking' - 'planning' and most importantly 'cooperating'.

4) Reduce in Size of Software Product
As the new technologies arrived in market various compressed techniques are there in the market. Even the size of the product is much lesser which can be compatible for the customer's point of view.

3.2 Old Way : Beginning of the Software

Fig. 3.2.1 : Old way for travelling

See the above picture in which a tempo travels on the road which is full of rocks and water. Tempo is heavily occupied with some containers and driver has overburden to drive the tempo till his destination.

If we compare the above example with software, it follows conventional Software project Matrix (SPM), and it includes the **Waterfall models** and **Software Management** performance which having various difficulties towards the achieve software goals.

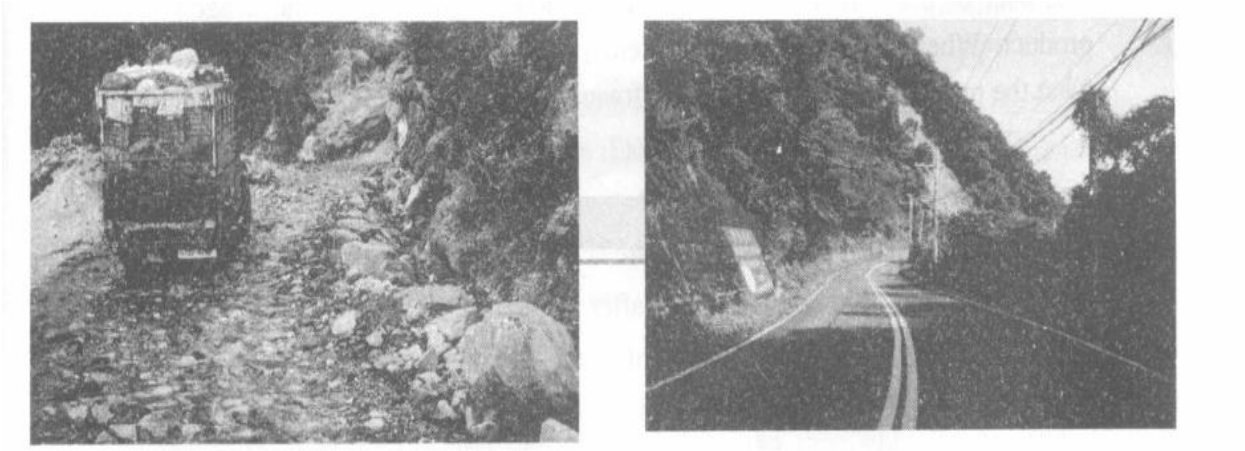


Fig. 3.2.2 : Comparison between old way and new way

As differentiation in above both figures, we make some points as

• Old roads are full of rocks while new are smooth plane.

• Old roads are having water and hence sleepy area while new roads having provision of the water circulation.

• New roads having indicators to destination which cannot in older roads.

• Various provisions such as street lights, road divider are available with the new roads.

• Obviously driver chooses the new way for travelling.

3.3 The Principles of Conventional Software Engineering

Following are the principles of the conventional software engineering, which are applicable to software development life cycle phases which encompasses of the software.

1. Focus on Quality:

Qualitative software always achieve the goods goals on the customer side. The teams involved in the designing the software are always have some goals. Team is made up of several members, which can have different quality aspects. The software can be made qualitative by making use of the qualities of each of those members in parallel way. If the lack of skilled members is used in the making of the software then it might degrade the software quality. In that case hiring the skilled quality team members are essential. Involving customer is another necessary, since finally he is user of the software. What is right and what is wrong is considered by his point of view is very essential. Likewise the team

can inspect the project the goes in smooth handling or not? The prototype used for that are place better or not? Project has simplified design or not? These are some question which gives the high quality software.

2. Role of the Customer:

End user of the software is always the customer who handle and usage the software product. When the customer can interact earlier with the software product, he judge what the problems before and after software product. He thinks on points;

- Time duration are reduced or not?
- Cost usages are reduced or not?
- How automation comes in picture?
- What kind of relaxation he has after software is in his hands?
- Which situation is better, current or earlier?
- Is the workload is really minimize? And so on ....

Answers of the all above questions are put by customer if and only if when he plays the software earlier.

1. Quality Software : Various parameters are there by which software maintains its quality. For example, we had taken in above point, which is involvement of customer in the quality consideration of the software. Other parameters are simple design, taking time to time inspection and appoint quality personals.

2. Finding the Problem : As the old way is fully depends upon the requirement and the model (water fall model) which we used in the old way is also called as 'Requirement -Driven Model'. But also defining the problem before the collecting and writing the requirement is very essential since most cases solution as per requirements are there but might be that solution can be in longer way. Since whatever alternative solutions are there that are also comes into the picture. And this is done only when 'exact problem is front of the software team '.

3. Model Designing : When architecture can design the building, before to that he make plan and afterword he follows that plan for make successful construction. The same process here in the software modeling that means project has some rules and regulation that are binds with some corporate culture. Ready to accept the risk factors.

4. Solution with alternatives way : When we discuss about the ways there are lots of ways from source to destination and traveler always thinks that which way is fruitful for him. Here the meaning of the fruitful is that safety, short distance, less costly etc. Likewise in case of the software designing, it cannot bind with state forward way it has some alternative. That entire alternative must be proven at the time of writing algorithms.

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

5. Various Languages for different functionality : There are so many solutions available today which converts the complex and difficult task to easier way. Various different principles are used to solve such complex problem. But user has satisfied is the last motive.

6. Closerto real world problems : Whenever all the real problems are satisfy then and then only real world problems are solved. This is done by minimizing intellectual distance.

7. Right best than Rapid: Rapid solution is always happy to customer. He thinks how his work rapidly goes on. But parallel to that it is also check that the job done is right way or not.

8. Code Checking : Software testing is more concern about the user satisfaction More sophisticated way is that inspecting the overall design and codes for finding the errors within it.

9. Management v/s Technology : As we compare management and technology, technology requires the lots of resources then and then only it goes smoothly, best technology cannot compensate for poor management. But the quality of good manager is handling to poor management also in efficient way for producing the best results. Best manager handle his overall team to motivate the job. In other words the best management is always better than best technology.

10. Hum an resources are Success key : The skilled work is not judged based on the proper and sufficient tools, technology desire language and the process which is used. It is based on the skillful person who knows the exact solution for the problem. This is not happen with the unskilled person or we can say that wrong person even if we provide him best technology, proper tool etc.

11. Curiosity handling : Whatever happening is might be correct, the answer may be appropriate, but if we cannot handle the processing very careful as per the environment which is not beneficial.

12. Responsibility Handling : As leader taking initiative in the managerial work is the most important task for the Management, like that at the time of the software designing, coding engineer must be take initiative in taking the responsibility.

13. Know the Customer Satisfaction : The final product is handling by the customer. He is the end user of the product. What is actual his need, is very essential fact in case of the designing that product. When customer makes confirm with that product then and then only product designer has make success. In this case at the initial stage it is important to that customer understand the customer priorities. The late delivery of product is might be good thing but the earlier delivery of product with wrong or less functionality is always wrong.

14. Innovations are always welcome : As per user requirements product are done, is ok with the user satisfaction. But word happy is more fruitful as compare with word satisfied. Whatever more functionality with new innovations is provided to the customer he is wants that much performance via the product.

15. Change become slowly : The new innovations are not directly accepted by the customer. If the product is new brand with variety of innovation the customer have thought he is in risk to accept that product. He don't know about that product, whether that product is really good or not. So there must be follow some basic facts which are customer previously known. Afterword new applications or algorithms may be produce with that only. Since new things may not be easy at the first time.

16. Design for change : As we renew our house, when there is function in house. As we feed of that renovation like that the architectures, components and specification techniques must be change within the software.

17. Essentiality of Documentation : Suppose one of the civil engineer complete the work of the bridge on the other road and after finishing the work he said that "I done bridge, now I see by paperwork that bridge has really good condition with that much columns". This is funny statement (Too much risky) since paperwork as drawing, checking is essential before to building that bridge. The height of that bridge might be passing down the truck.

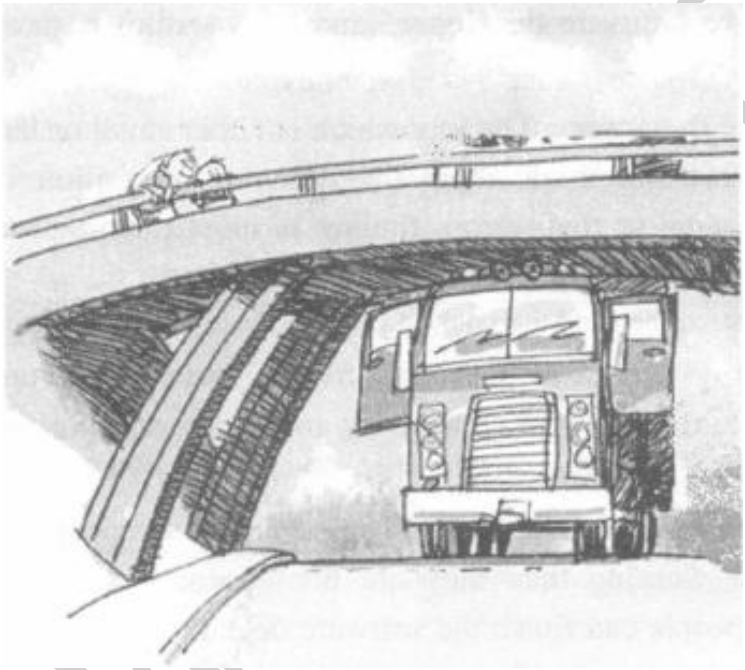


Fig. 3.3.1: Problematic bridge for large vehicles

Same to that, designing completed without documentation is not the proper work at all.

18. Tools for its practical: By using software project customer has comfort in his work is a fact. Like if we are use Microsoft PowerPoint for mock presentation then it is reliable, we will use word for typing the book.

19. Ticks - less always better: At the time of self learning tricky programming is very essential for logic building. Some of the programmer are love to make tricky programming. But at the customer point of view, tricks are not better even if it is better for programmer or designer to do smart logical programming with less tricky codes.

20. Encapsulation: Encapsulation is the wrapping the several material in single unit. By using that information to be hidden in simple way.

21. Coupling and Cohesion: What kind of maintainability and adaptability, inherits in software measured by the coupling and cohesion.

22. McCabe complexity measure: Even though there are many number of metrics available to track and report the inherent complexity of software, but none of them is as intuitive and easy to use as Tom McCabe's. Therefore use of McCabe complexity measure is insisted.

23. Test by Others: As we know customer is end user of the software, he puts his requirements in front of the software engineer. Software engineer afterword makes a plan for designs and developing. Once software completes then it is essential to test it to various different reasons. If we design and develop the software then we cannot test it own since software developers cannot test their own developed software. The reason is end user can handle software, whatever problems he face earlier is minimize or not, is cannot judge by the software developer, since that testing is must done by the other person.

24. Find out Cause of the error : The bug which is either small or large, removing that is very essential. When we think about the thought 'prevention is better than cure', software logic is same to that, errors finding is most important task since, when we analyze them and prevent them it is cost effective method. W hen error is detected then and then only it is fixed, since this is analyzing method.

25. Software's entropy increases : Any software system that undergoes continuous changes in it will surely grow in complexity and becomes more and more unorganized and abundant.

26. No relation between People and Time : If we consider the people, who are engaging in the completion of working then they are not depends on the time parameter. For example, if three people can finish the software development in one month, that cannot indicates that, they have ready 12 projects in one year. In other words we say that, time and people are not interchangeable.

27. Expert excellence : We can produce good jobs (products) from our team. Whatever final success is totally depends on our expertise team. Since here when we have more expectation from our employee then and then only our employee makes better work.

3.4 Principles of Modern Software Management

As per Davis format there are 10 principles of the modem management. These principles are placed in priority-wise as below;

1. Architecture First Approach : The design architecture primarily and the lifecycle plan. Afterword resources should be finalized for full scale development.

2. Build iterative life cycle process : In the latest software industry it is not possible that fixed whole problem, as per that problem generate designs whole, make entire software and finally test it that means whole in proper sequence. In the iterative process problems can be rectify previously and give proper solution for that. Most of the risks are known earlier, which is useful for increasing predictability and use to avoid expense of repetitive work.

3. Component Based Development : As custom development is possible rather than human generated source code then it is possible to move on line of code generation to component based code generation. Since component consists of executable format predefine source code.

4. Change Management Environment : Change in every fact is essential, it is gives the dynamic iterative model. If we use different - different peoples or teams for the same job then that is workflow follows the change management environment.

5. Round Trip Engineering : Automation in every sector of software is very essential since without to that automation bookkeeping, changing, designing, documentation, coding and finally testing is difficult. In iterative process change liberty is essential. These are the top 5 principles in software management. Following figure depict it in brief idea.

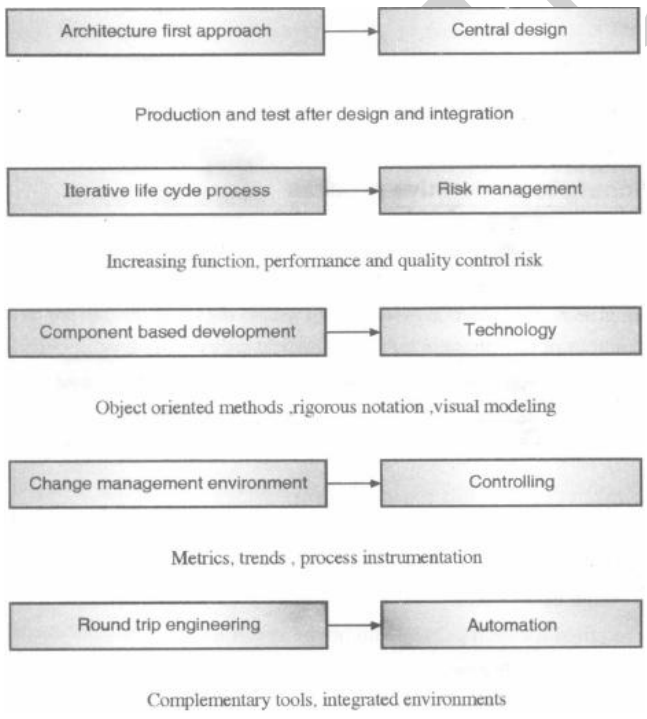


Fig. 3.4.1 : Principles of software management

(i) Model Based notation

Develop design artifacts using model-based notations. This makes the reviewing easy compared to the inspection process.

(ii) **Objective Quality Control**

The quality can be controlled by assessing the progress. Once the assessment is over, it is integrated with the development process.

(iii) **Demonstration based Approach**

This approach is used to assess intermediate artifacts. This approach is generally applied to early prototypes, baseline architectures, and early releases.

(iv) **Evolving levels of details**

Using these evolving levels of details, the intermediate releases are planned so as to allow early and continuous releases. And these release are accompanied with corresponding use-cases and scenarios.

(v) **Configurable Process**

A configurable process is adopted that is economical and is scalable across a wide range of projects.

3.5 Transitioning to an iterative process

First of all we have a question i.e. why waterfall model (which is traditional model in the software engineering) is no longer used? The answer is that, in waterfall model, each phase depends on the previous phase. It indicates that, the output of one phase is input for the next consistent phase. The latest modern processes depend upon the initial version of the system. E.g. Spiral, Increments, Generations, Release.

Drawbacks of the Waterfall Model are as follows :

1. Real products rarely follow this sequential flow.
2. It is very difficult for customer to state all the requirements in one time.
3. Many projects face this requirement uncertainty at beginning itself and so it is very difficult to design next phases.
4. Time span required for each phase could not be specified.
5. Naturally project requires more time.

6 . Project becomes lengthy also.
7. Customer should have patience.

Iterative Process :

• It can also be called as *Incremental Process Model* and the basic idea is that the software is developed in increments, where each increment adds some functional capability to the system until the full system is implemented.
• In iterative enhancement, extensions and design modifications in the project can be made at each step.

Need of Iterative Process :

• The *iterative life cycle model* removes the limitations of the waterfall model and tries to combine the benefits of both prototyping and the waterfall model.
• The iterative process models combine the elements of waterfall models applied in iterative fashion.
1. It adapts all the phases of waterfall model.
2. It accepts linear process.
3. It tries to solve one by one problems of the customer, or it accomplishes one by one requirement of the user, which is called as incremental iterations.
4. Generally first increment is nothing but core product of the customer.
5. As time grows, it is incremented from one requirement to next requirement or simultaneously it performs both the requirements (i.e. old requirement as well as starts new requirements)

Advantages of Iterative Process :

1. Allows early development of initial versions.
2. Risk areas are addressed early in project life cycle phases.
3. Several iterations are developed.

Examples : Incremental model, Rapid Application Development model (RAD), Spiral model

1. It can result in better testing, since testing each increment is likely to be easier than testing entire system like in the waterfall model.
2. As in prototyping, the increment provides feedback to the client, which is useful for determining the final requirements of the system.

Review Questions

Q. 1 Discuss the parameter which are use for economic development.
Q. 2 How convensional software model focuses on Quality ?
Q. 3 Explain Role of customer in conventional software engineering.
Q. 4 Explain S principles of modern software management ?
Q. 5 What are drawbacks of waterfall model ?
Q. 6 Why iterative process is needful ? What are its advantages.

4. Life Cycle Phases

Syllabus :
Engineering and Production Stages, Inception, Elaboration, Construction, transition Phases

4.1 Software Process

A software process identifies a set of activities that are applicable to the development of any software project, regardless of their size or complexity. A software process is a collection of work activities, actions and tasks that are to be performed when some software project is to be developed.

Software process can be categorized into :
1. Generic Process model - represents a framework activity populated by a set of software engineering activities.
4. Personal and Team Process models - This model helps in creating a software that best fits either the personal needs of the user or that meets the broader needs of a team.
3. Prescriptive Process models - provides an ordered structure and an effective roadmap to software engineering work.

4.2 Generic Process Model

This model defines a set of *umbrella activities* which are also a must for any software engineering process as shown in the below figure.

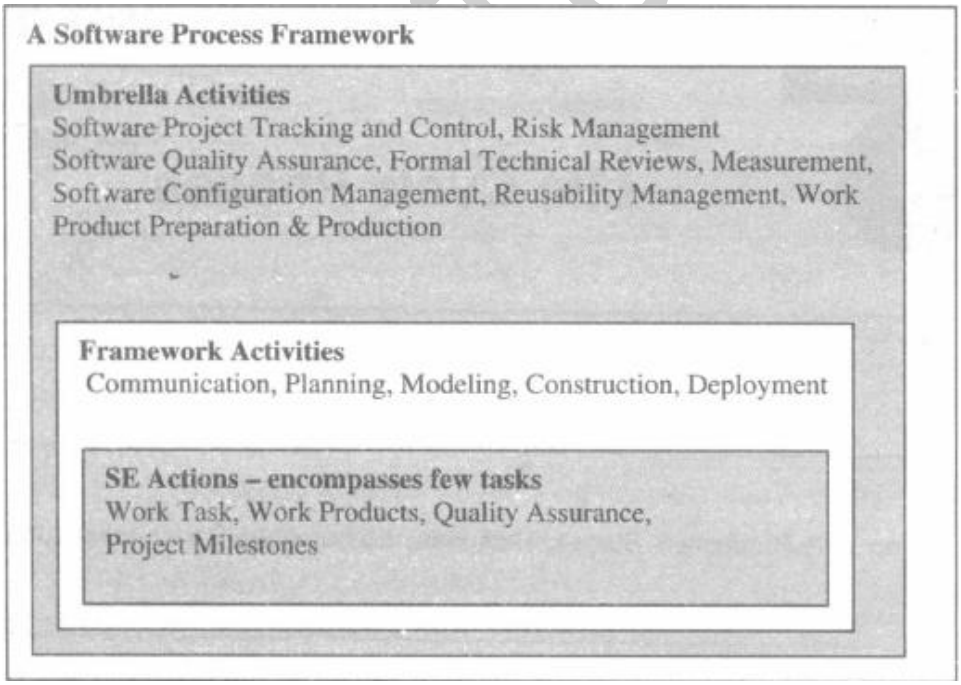


Fig 4.2.1 : A Software Process

Again, each framework activity contains a set of software engineering activities which is a collection of tasks that develops a major software product.

4.2.1 Framework Activities

First, we take a look at the generic process framework activities that are must for the development o f any software project:

• **Communication :** The main cause of communication is requirement gathering. This activity establishes sufficient interaction or collaboration between the developer and the customer for gathering the requirements and knowing the expectations of the customer.

Example: we can explain the **work task** regarding the **communication activity** of a simple project as listed below :

o Making the list of the end-users, software engineers and support people for the project.
o Inviting all of them for an informal meeting,
o Ask each end-user to make a list of features and functions required,
o Discuss these requirements and prepare a final list,
o Arrange the requirements according to their priority,
o Note the areas of uncertainty.

• **Planning :** This activity defines the software development process to be conducted. It describes all the needed technical tasks, possible risks, the resources that are required, the work product to be produced and the schedule to workout the whole process. This activity plans the work, identifies the resources, tasks and sets the schedule.

• **Modeling** : This activity creates a model (blueprint) which clearly describes the software requirements and the design that will achieve these requirements. This is helpful to both customer and the developer respectively, to understand what he wants from the software and how he can develop it. Modeling is composed of two main activities-*analysis* (requirements gathering, elaboration, negotiation, specification and validation) and *design* (data design, interface design and each module level design).

• **Construction** : This activity includes code generation either manually or using automated tools and then testing the code to correct the errors if any.

• **Deployment** : The software (as a complete product or in a partial stage) is delivered to the customer who then checks the product and provides feedback on evaluation. *The framework activities are applied on every project but the degree of tasks depend on the :*

o Type of the project
o Characteristics of the project
o Agreement of the project team on common views.

4.2.2 Process Iteration and Activities

The above framework activities discussed in sec 4.2.1 occur in an organized pattern with respect to sequence and time. This work flow pattern of the activities is termed as *'Process Flow'*

1. Linear Process Flow : Executes the five framework activities in a sequence starting with 'communication' and ending with 'deployment'.

Fig. 4.2.2: Linear Process Flow

2. Iterative Process Flow : Repeats one or more of the five framework activities before proceeding to the next.

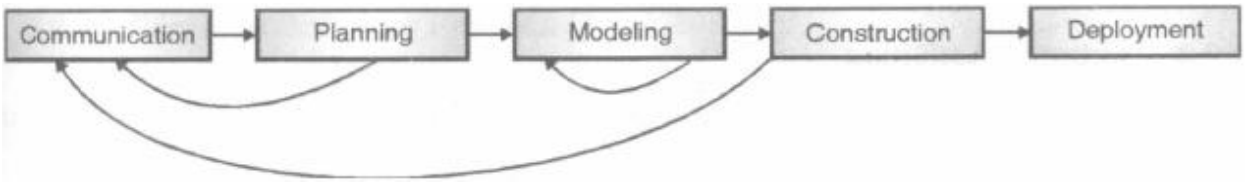


Fig. 4.2.3 : Iterative Process Flow

3. Evolutionary process Flow : Executes the five framework activities in a 'circular/cyclic' manner.

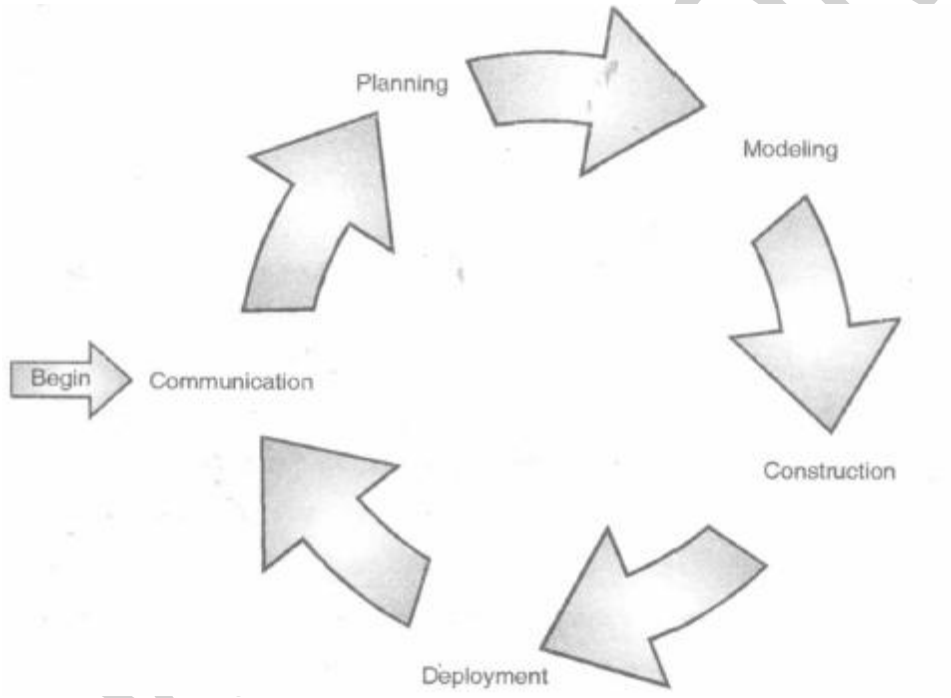


Fig. 4.2.4 : Evolutionary Process Flow

4. Parallel Process Flow : At a time, executes one or more activities i.e. one or more of the five framework activities are executed in parallel with the other. Say, modelling of one module is executed parallel to the construction of another module.

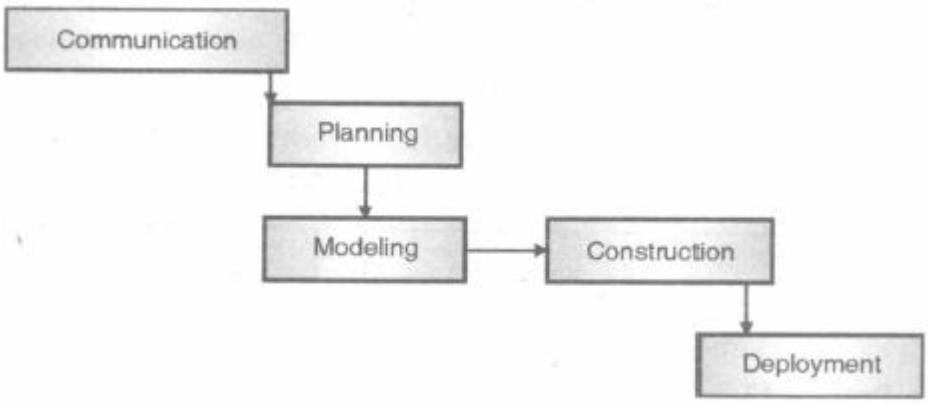


Fig. 4.2.5 : Parallel Process Flow

4.2.3 Umbrella Activities

Now, we look at the **Umbrella activities** that are applicable throughout the software process :

• **Software Project Tracking and Control** : Software team assesses the progress of the project plan time to time and takes necessary action to maintain the schedule. Thus, software team tracks and controls the project schedule.

• **Risk Management**: Software team assesses the risks that may affect the outcome of the project or say the quality of the product.

• **Software Quality Assurance** : Software team defines and conducts the activities needed to preserve the quality of the software product.

• **Formal Technical Reviews** : Software team assesses the technical efforts to find and remove the errors before they are forwarded to the next action.

• **Measurement** : Just the coincidence is the four P 's of Software Engineering : Project (the task at hand), Process (the manner it is done), Product (the object produced) and People (by whom it is done). Software team collects all the project, process and product measures so that it can be used in combination with all other framework and umbrella activities.

• **Software Configuration Management**: Software team takes essential steps to manage the affects of changes made throughout the software process.

• **Reusability Management** : Software team defines criteria for work product reuse and establishes mechanisms to achieve reusable components.

• **Work Product Preparation and Production** : Encompasses the activities required to create work products such as models, documents, logs, forms and lists.

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

4.3 Personal and Team Process Models

• The whole *process* of developing a *product* throughout the *project* is handled by the *people* working on it.
• *The software process can either be **personal** or **team** built.*
• Some small hardware or software products can be developed by individuals (personals), but the scale and complexity of modem systems is such, and the demand for short schedules is so great, that it is no longer possible for one person to do most engineering jobs. Systems development is a team activity, and the efficiency of the team largely decides the quality of the engineering.

4.3.1 Personal Software Process (PSP)

• The PSP provides engineers with a disciplined personal framework for doing software work.
• This personal framework of PSP model consists of five main activities :

1. Planning:

This activity isolates and defines the requirements and based on these, develops the size and resource estimates and also identifies the probable defects. All these metrics are recorded on worksheets, first they are analysed and then finally the development tasks are identified and the project schedule is created.

2. High level design :

This activity identifies the external specifications needed to construct each component and accordingly the design is created. And when there is uncertainly, the prototypes are built.

3. High level design view :

This activity performs the verification methods to find and remove the errors in the coding or design.

4. Development:

Every module (component) design is reviewed in detail. Accordingly, the corresponding code is also generated, reviewed, compiled and tested so as to achieve 'zero-defect' product.

5. Postmortem:

All the four activities described above are controlled, tracked and recorded time to time properly on the worksheets. Using the information collected from the above activities, the effectiveness and the quality of the software development process is determined. This information helps in suggesting any changes in the process, if required, so as to improve the software quality.

The PSP helps software engineers to :
- manage the quality of their project
- make commitments they can meet
- improve their estimating and planning skills

- reduce the defects in their work and ultimately in their product. This is achieved through a rigorous assessment of all activities done by the software engineer.
• The goal of PSP is to help developers produce quality and zero-defect products on time. For example; Motorola division in Florida achieved zero defects in over 18 projects through implementing PSP technique.
• The PSP is a prerequisite for an organization i.e. planning to introduce TSP.
• The PSP can be applied to many parts of the software development process such as :
- small-program development
- requirement definition
- document writing
- systems tests
- systems maintenance
- enhancement of large software systems

4.3.2 Team Software Process (TSP)

The common definition for a team :
• A team consists of at least two people.
• The members are working toward a common goal.
• Each person has a specific assigned role which avoids conflict between the team members, avoids duplicate work and time wastage.
• Completion of the mission requires some form of dependency among the group members. Each team member depends to some degree on the performance of the other members. Interdependence improves individual performance because the members can help and support each other.
The goal of TSP is to build "self-directed" project team that organizes itself to produce high quality software. The objectives of TSP are to :

• Build self directed teams that plan and track their work, establish goals and own their processes and plans. These can be pure software teams or integrated product teams of 3 to about 20 engineers.

Provide a simple process framework based on the PSP.
Show managers how to coach and motivate their teams to sustain peak performance.
Use modest, well-defined problems.
Develop products in several cycles.
Establish standard measures for quality and performance.
Provide detailed role definitions.
Use role and team evaluations.
Require process discipline.
Provide guidance on teamwork problems.

First version of the TSP process was developed in 1996 by Watts Humphrey. His objective was to provide an operational process to help engineers consistently do quality work.

TSP helps the engineers to :
• ensure quality software products
• create secure software products

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

• improve process management in an organization

Each project is launched using a sequence of tasks that enables the team to establish a solid basis for starting the project. The TSP launch process includes below tasks :

• establish product and project goals
• define and assigning team roles
• assess risks
• develop the quality plan and set quality targets
• plan for needed support facilities
• produce an overall development strategy
• make a development plan for the entire project
• make detailed plans for each team member for the next phase
• merge the individual plans into a team plan
• rebalance the team workload to achieve a minimum overall schedule
• Assess project risks and assign tracking responsibility for each key risk.
• After the launch, the TSP provides a defined process framework for managing, tracking and reporting the team's progress.

4.4 Prescriptive Process Models

Definition : Prescriptive models define a discrete set of activities and actions to accomplish all tasks of the software with milestones, which is used to develop the software. These Process models may not be perfect but they give very good guidance in software development process.

Uses
It is used by
1. Software engineer.
2. Manager.
3. All employees who play important role to develop the software.

Importance
This model is important because,
1. It provides stability
2. It provides control for well organization activity.
3. It is also referred as rigorous model.

Steps
This model takes following steps :
1. This process guides software team.
2. It generates frame work activities to organize into a process flow.
3. Process flow may be linear or incremental or evolutionary.
4. The terminology of each model is different.

Work product

The work product is the programs, documents and data that produce a sequence of activities as well as task defined by the system.

Ensure
This mechanism determines the maturity of the software using quality, timeliness and long term validity of the product.
The best indicators are the users who use the product and judge the efficiency.
The prescriptive model also called as a conventional process model. In short...

1. It describes a unique set of frame work activities.
2. It should populate framework activities to set a software engineering action.
3. These actions are used to create work product to accomplish to meet development goal.
4. It finds out- the nature of project, whether is suitable for the people using it, whether it is suitable for the environment, where it is implemented.
5. Framework activities are communication, planning, modeling, construction, and deployment.
6. It prescribes a set of process elements, framework activities, software engineering actions tasks, work products quality assurance, change control mechanism of each project.

Examples: *The various prescriptive (conventional process) models are waterfall model, incremental model, rapid application development model, prototyping model and spiral model.*

4.5 Life Cycle Phases

• Mostly, *all software development process* overemphasizes *either* on: i) research and development *or* on ii) production.
• But, characteristic of a *successful software development process* is achieved by a proper separation between i) *"research* and *development"* ii) *activities* and *"production"* activities.
• Projects developed under the conventional process have a very precise project milestone when there is a transition from a research step to a production step. The initial phases focus on achieving functionality and the later phases focus on achieving a product that can be delivered to the customer, with special attention on the product's robustness, performance, and finish.
• Similarly, the modern software development process must be defined to support the following in-order to achieve successful software development process:

o Designing the plans, requirements, and architecture, together with well defined clear and precise points,
o Managing the risks and measuring the objectives to achieve quality.

• Developing the system capabilities by increasing the system functionalities.

4.5.1 Engineering and Production Stages

To achieve higher returns on investment, there is a need to adopt such a software manufacturing process which is driven by technological improvements in process automation and component-based development.

The two particular stages of the life cycle are:

1. The **engineering stages** are less predictable and have smaller teams involved in design and synthesis activities. It focuses on risk reduction, prototyping, establishing architectural baseline, analysis, design, and planning.
2. The **production stages** are more predictable and have larger teams involved in construction, testing, and deployment activities.

Table 4.5.1 : Two Stages of the Life Cycle - Engineering and Production

Life-Cycle Aspect	Focus of Engineering Stage	Focus of Production Stage
Risk reduction	Schedule, technical feasibility	Cost
Products	Architecture baseline	Product release baselines
Activities	Analysis, design, planning	Implementation, testing
Assessment	Demonstration, inspection, analysis	Testing
Economics	Resolving diseconomies of scale	Exploiting economies of scale
Management	Planning	Operations

Engineering stage is carried out in two distinct phases - *inception and elaboration* Similarly, Production stage is also carried out in two distinct phases - *construction and transition.*
All these four phases of the life-cycle process are loosely mapped into a conceptual framework of the spiral model as shown below;

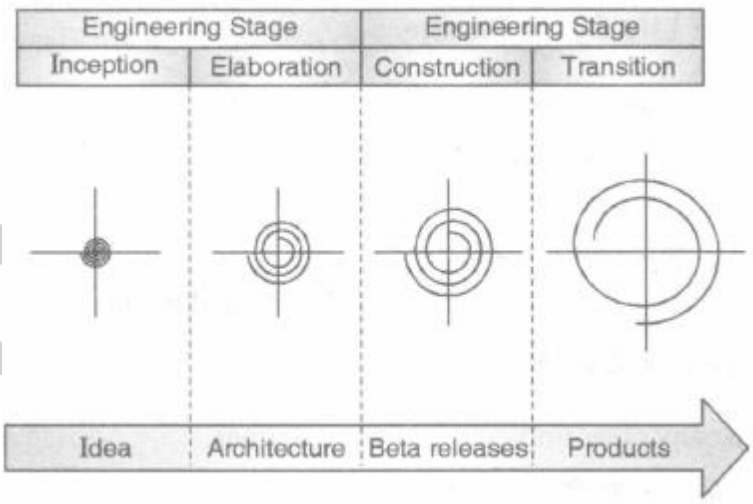


Fig. 4.5.1 : Four Phases of a Life Cycle Process

These life cycle phases are very widely practiced and adopted in the industries.

4.5.2 Features of Life Cycle Phases :

• The most important feature/idea is *Iterative Development.* Iterative Development is sequentially expanding and refining a system through multiple iterations, using feedback and adaptation.

• Each phase has iterations, each having the purpose of producing an executable piece of software. The duration of iteration may vary from phase to phase.

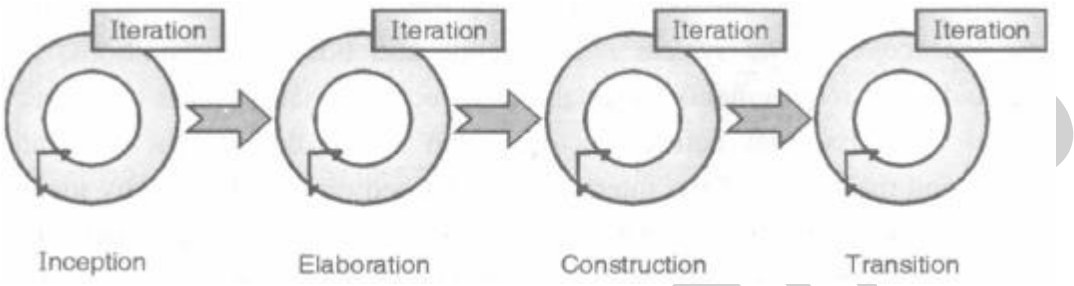


Fig. 4.5.2 : Four phases of Iterations

• It is a lightweight process addressing the needs of small projects - to more comprehensive process addressing the needs of large projects.

• It helps in early and continuous documentation of the most urgent and the most probable risks by proper planning and keeping a follow up. This helps in mitigating the risks at early phases of software development.

• It also uses visualization methods such as an UML to build models so as to understand the complexity of the system.

• It also uses *use-cases* as test cases which allows end-user documentation and helps in designing.

4.5.3 Advantages of Life Cycle Phases :

It emphasizes on addressing very early high risks areas.

It does not assume a fixed set of firm requirements at the inception of the project, but allows refining the requirements as the project evolves.

It does not put a strong focus on documents.

The main focus remains the software product itself, and its quality.

4.5.4 Drawbacks of Life Cycle Phases :

It fails to provide any clear implementation guidelines, leaves the tailoring entirely to the user.

4.6 Inception

Inception means *start* i.e. this is the point where the project is proposed.

Goal of Inception :

• To achieve concurrency among the stakeholders of the system on the objectives decided for the project.

Objectives of Inception Phase : It constitutes of Business modelling i.e .:

• ***Define the problem*** : The objectives of the project are stated, so that the needs (requirements) of every stakeholder are considered.
• ***Define the scope of the system :*** The Scope and boundary conditions, acceptance criteria and some requirements are established such as what is and is not expected to be in the product. External entities (actors) with which the system will interact are identified and the nature of the interaction is defined on a high-level by identifying all use cases. It also includes identifying the business case i.e. identifying the success criteria, risk assessments and estimation of the resources needed and a phase plan showing dates of major milestones.

• *Separate the vital use cases of the system* and the main scenarios of the operation that may drive the major design trade-offs.
• Demonstrate at least one candidate-architecture against some of the primary scenarios.
• *Estimate the cost and schedule* needed for the entire project.
• *Estimate the potential risks* i.e. the sources that may cause unpredictable problems.
• *Initiating the Project:*And then, you can start working on the project.

Activities in Inception Phase :
• To define the project scope: The scope of the project is defined in the SRS (Software Requirement Specification). This also defines the problem and derives the acceptance criteria for the proposed product.
• To develop the architecture: The SRS must reveal the feasibility of at least one candidate-architecture and also must define an initial baseline of decisions so that the required expenditures (cost), schedule needed for development and delivery, and required resources can be estimated.
• To plan and prepare a business case : This involves the evaluation of the alternatives proposed for risk management, staffing, iteration plans, cost,/and schedule/profitability.

Evaluation Criteria :
• Do all stakeholders agree on the defined scope and cost and schedule estimates?
• Are requirements precise, clear and understood as defined in the use cases?
• Are the cost and schedule estimates, priorities, risks, and development processes realistic?
• Does the depth of an architecture prototype define the preceding criteria? i.e. does it provide a vehicle for understanding the scope and assessing the efforts of the development group in solving particular technical problem.
• Does actual resource expenditure match approximately with planned expenditures.

Feasibility Study in Inception Phase :
• A new project is started when a *new business need is identified* or a new service is discovered. Stake holders of the business (business managers, marketing people, and product managers) define a business case for the idea, try to identify the breadth and depth of the market, do a rough *feasibility analysis*, and identify the *project scope.* All this information is enough to discuss with the software engineer to *start a project.*

• The feasibility study made by stakeholders can be as below which decides whether or not the proposed system is useful. The study checks :

o Will the system add to organisational benefits ?
o Can the system be engineered using current technology and within budget ?
o Can the system be integrated with other systems that are used ?

Interviewing the Customers and Users in Inception Phase :
At project Inception time, software engineers ask some questions to the people in the organisation based on the information collected :

• What if the system wasn't implemented ?
• What are current process problems ?
• How will the proposed system help ?
• What will be the integration problems ?
• Is new technology needed ?
• What facilities must be supported by the proposed system ?

The need of these questions is to find :
o The effectiveness of collaboration between the customer and developer,
o Basic understanding of the problem,
o Who will use the solution ?
o The desired nature of the solution.

Inception Outcome :
• Vision document
• Initial use-case study (10%-20% complete)
• Initial business case
• Initial risk assessment
• Project plan
• Stakeholders decide whether to commence a full scale project or not.

4.7 Elaboration

• Elaboration means *refinement* (careful development). This phase is the end of "engineering" stage.
• In this phase, an executable architecture prototype is developed in one or more iterations depending upon the scope, size, & risk.
• Elaboration is about creating an analysis model that defines the informational, functional and behavioural aspects of the problem.
• The main task is to describe the problem in a way that establishes a firm base for designing a model.
• Elaboration focuses on *expanding* the information (i.e. obtained from inception and elicitation) and then developing a *refined* technical model of software functions, features
and constraints (i.e. restrictions or limitations). This process is composed of various modelling and refinement tasks.
• Different models may be produced during this activity depending on the relationships and collaboration between the various business domain entities.

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

• From the designed model, it would be easy to judge if the efficiency of workflow of the system is as it has been imagined.

Goal of Elaboration Phase :
• This phase gives you a *mile wide and inch deep* view i.e. little bit deeper view of the system.
• Detailed analysis of the problem resulting in the definition of an architectural foundation for the project. It constitutes of *requirements, analysis* and *design* phases.
• This phase ensures that:
o the architecture, the requirements, and the plans are defined and there is no change now;
o the risks are traced and fixed;
o the schedule for completion of development is predicted within an acceptance criteria.

Objectives:
• baselining an architecture as rapidly as practical
• baselining a vision
• baselining a sound plan for the construction phase
• Representing that the baseline architecture supports the vision at a reasonable cost in a reasonable time

Activities in Elaboration Phase :
• Detailing the vision.
• Detailing the process and infrastructure.
• Detailing the architecture and defining the required components.
• Eliminate the highest risk elements of the project.

Evaluation Criteria :
• Is the defined vision stable?
• Is the defined architecture stable?
• Are the major risk elements eliminated from the process?
• Is the construction phase plan reliable and are the required estimations done?
• Do all stakeholders agree that the current vision can be achieved by following the current plan for developing the complete system in context of the current architecture?
• Does actual resource expenditure match approximately with planned expenditures.

Elaboration Outcome :
• Use-case model will be 80% complete.
• Additional requirements capturing the non functional requirements and requirements not associated with a specific use-case are identified.
• Description of the Software Architecture.
• An executable architectural prototype is developed.
• A revised risk list and revised business case is developed.
• A development plan of the whole project. This defines the iterations and evaluation criteria needed for each iteration and also the process i.e. to be used.

4.8 Construction

Construction means *to build* i.e. it is a manufacturing process. *Aim of Construction Phase* : It constitutes of implementation i.e. detailed design and construction of source code.

Objectives of Construction Phase :

• To minimize the software development cost by reducing the required number of resources and avoiding the unnecessary rework.
• To achieve enough software quality as fast as possible.
• To keep track of program versions (such as the alpha, beta, and other test releases) as fast as possible.

Activities in Construction phase :

• Emphasizes on managing resources and controlling operations to optimize costs, schedules and quality. This phase is broken into several iterations.
• In this phase, all the remaining components and the remaining application features are integrated into one application and then all the features are thoroughly tested after the integration.
• Assessing the product releases against acceptance criteria of the vision

Construction Outcome:

• An executable product that is ready to put in the hands of the end users.
• The software product integrated on the adequate platform.
• A user manual.
• Description of the current release.
• This is considered as a *beta* release.

Evaluation Criteria :

• Is the product baseline stable enough so as to be deployed on the user side?
• Are the stakeholders ready for transition of the product on to the user side?
• Does actual resource expenditure match approximately with planned expenditures?

4.9 Transition

Transition means *delivery*. The transition phase is the phase where the product is put in the hands of its end users. This phase is entered when the product is mature enough to be deployed on the enduser's site.

Goal of Transition Phase :

It constitutes of deployment i.e. delivery of the system to the user community. It involves issues of marketing, packaging, installing, configuring, supporting the usercommunity, making corrections, etc.

Activities in Transition Phase:

• Deliver the software product to the user community.
• Issue new releases

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

• Correcting problems (if any)
• Finish the features that were postponed
• Perform beta-testing to validate new system against user expectations
• The system might run in parallel with a legacy system that it is replacing
• Training of user maintainers
• Roll-out the product to marketing, distribution, and sales team

Objectives:
• Achieving user self -supportability
• Achieving stakeholders' agreement that deployment is complete and consistent.
The transition phase ends when the deployment of the product achieves the complete positive agreement of the customers and users.

Evaluation Criteria :
• Is the customer and user satisfied?
• Are actual resource expenditures versus planned expenditures acceptable?
• Does actual resource expenditure match approximately with planned expenditures?

Transition Outcome:
• Matured enough product that can be deployable on user's site.

4.9.1 Six Best Practices

Life Cycle Stages is built on the "Six Best Practices" :
1. Develop iteratively :
• Software must be developed in small increments and short iterations.
• An iterative process breaks a development cycle into a *sequence o f 4 phases* each of which includes a *series o f iterations.*
***2*. Manage requirements :**
• It allows accommodating requirement changes in system development strategy.
• Those requirements that change over time and those requirements that have greater impact on project goals are identified.
• It is a continuous process to identify requirements.
• Managing requirements include :
o Elicit, organize (according to the priority), and document the required functionalities and constraints,
o Evaluate the impact of changes and
o Track and document the decisions.
3. Use component architecture :
• The process focuses on early development and design of independent executable modules, prior to committing for full-scale development.
• Components that are most likely to change and components that can be re-used are identified and built.
4. Model visually :
• Models must be built using visualization methods like that of the UML, to understand the complexity of the system.

• This helps you to understand the different aspects of your software and see how the different elements of the system communicate with each other.
• Maintains uniformity between design and its implementation.
• Promotes unambiguous communication between developer and end user.

5. Verify quality :

• Quality of the software is maintained by its frequent testing.
• Testing is done to remove defects at early stages, thus reducing the cost at later stages. In particular, high risk areas are tested more thoroughly.
• The software released at the end of every iteration is tested and verified.
• Test cases are created based on use cases (and its scenario).
• Decisions are made on real test results.

6. Control changes :

• Any changes to requirements must be managed and their effect on software should be tracked.
• All change control goes through the convener of the CCB (Change Control Board).
• Members of CCB can be representatives from different areas, say: test designer, project manager, system analyst or stakeholders.

Review Questions

Q.1 Define the software process.
Q.2 List out the Software Process Framework Activities.
Q.3 List out the Software Process Umbrella Activities.
Q. 4 List out the different life cycle stages and what happens in those stages.
Q. 5 Describe the four phase in the Life cycle
Q.6 Explain generic view of process.
Q. 7 Explain PSP and TSP.
Q.8 State the six best practices followed in the life cycle stages.

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

5. Artifacts of the Process

Syllabus :
Artifacts of the process: The artifact sets, Management artifacts, Engineering artifacts, programmatic artifacts.

5.1 The Artifact Sets

To manage the complete development process of a software system, there is a need to collect and organize the distinct sets of information. These organized distinct sets are known as *artifact sets.* *Artifacts* involve the cohesive information that is generally developed and reviewed as a single entity.
The information used in our Life-cycle model that we have studied till now can be organized in to sets called as software artifacts. It can be organized into *five distinct artifact sets that are partitioned* as follows :
1. **Management set** (ad hoc textual formats),
2. **Requirements set** (organized text and models describing the problem),
3. **Design set** (models of the solution space),
4. **Implementation set** (programming and associated source files), and
5. **Deployment set** (machine-process able languages and associated files).

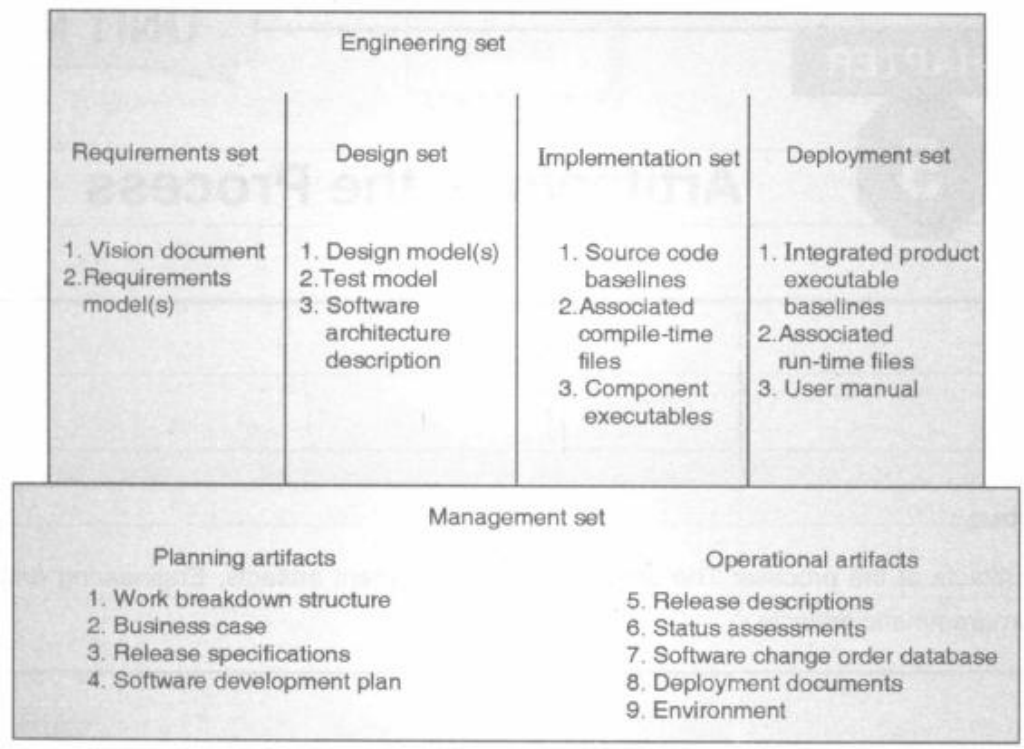


Fig. 5.1.1: Five Distinct Artifact Sets

5.1.1 Management Set

• The 'management set' involves the artifacts (information set) associated with process planning and execution.
• These artifacts use ad hoc notations, text, and graphics, or any other type of representations that are required to attain the "contracts" such as that of :
o Among the *project development team* (such as the project managers, architects, developers, testers, marketers, and administrators),
O Among the *stake holders* (such as financing authority, users, software project manager, organization manager, regulatory agency), and
o Between the project development team and stakeholders.
• Artifacts described in this 'management set' can also include the work breakdown structure for tracking :
o The activities and schedule
The *financial mechanisms* such as expenditures and profit expectations, The *release specifications* such as project scope, plan and objectives
The *software development plan* such as tracking of project process at any instance,
o The *status assessments* that may include the snapshots of project progress at any instance.
o The software change orders i.e. tracking the versions of changed software modules.
o The *deployment documents* such as the cutover plan and the training course
o The *hardware and software environment that include* software tools,process automation, and documentation.
• Artifacts of the 'Management set' are evaluated, assessed, analyzed and measured based on the:
o Stakeholder review
o Changes analyzed between the current version of the artifact corresponding to its previous versions
o Major milestone of the balance described among all artifacts and specially the accuracy of the business case and vision artifacts.

5.1.2 Engineering Set

The 'engineering set' includes four types of sets :

1.Requirements set,
2. Design set,
3. Implementation set and
4. Deployment set.

1. Requirements Set:
'Requirement set' artifacts are evaluated, assessed, analyzed and measured based on the :
• Consistency of the release specifications of the 'management set'
• Consistency between the vision and the requirements models
• Consistency, completeness and the semantic balance between information in the 'design', 'implementation', and 'deployment sets' derived from the mappings against these sets.
• Changes analyzed between the current version of the artifact corresponding to its previous versions (scrap, rework, and defect eliminations)
• Review of other quality factors

2. Design Set:

• The models in the 'design set' are engineered using the UML notations. These design models are essential for achieving the solution.

• The 'design set' consists of varied levels of abstraction that represent the components of the product solution such as the 'Class diagram' represents the component's identities, attributes, static relationships, dynamic interactions and etc.

• 'Design set' artifacts are evaluated, assessed, analyzed and measured based on the :

o The internal consistency and quality of the design model

o The consistency with the requirements models

o Translation of artifacts from design set into the implementation and deployment sets

o The consistency, completeness and the semantic balance between information in these sets

o Changes analyzed between the current version of the artifacts in design model corresponding to its previous versions (scrap, rework, and defect eliminations)

o Review of other quality factors

3. Implementation Set:

• The 'implementation set' contains the source code i.e. the programming language notations that describe the implementation pattern of the components such as their interface or the dependency relationships and etc.

• 'Implementation set' artifacts are written in human-readable formats which are evaluated, assessed, analyzed and measured based on the :

o Consistency with the design models

o Translation of artifacts from implementation set into the deployment set notations such as compilation and linking so as to evaluate the consistency and completeness among artifact sets,

o Executable files assessed against the relevant evaluation criteria. This assessment is done through inspection, analysis, demonstration, or testing,

o Execution of test cases that compare expected results with the actual outcome.

o changes analyzed between the current version of the artifacts in design model corresponding to its previous versions (scrap, rework, and defect eliminations)

o Review of other quality factors.

4. Deployment Set:

• The 'deployment set' contains :

o User deliverables

o Machine language notations

o Executable software product

o Build scripts

o Installation scripts and

o Executable target specific data that is required for using the product in its target environment.

• Deployment sets are evaluated, assessed, analyzed and measured based on the:

o Tests conducted against the user requirements and quality attributes so as to evaluate the consistency, completeness and the semantic balance between the artifacts in the two sets,

o Tests conducted on the partitioning, replication, and allocation strategies described in the mapping components of the 'Implementation set' against the physical resources of the 'deployment system' such as that of the platform type, number, and network topology,

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

o changes analyzed between the current version of the artifacts in design model corresponding to its previous versions (scrap, rework, and defect eliminations)
o Review of other quality factors
Each artifact set described above has a prime development focus matching with any one of the phases of the life cycle.
• Requirements set is the focus of the inception phase;
• Design set is the focus of the elaboration phase;
• Implementation set is the focus of the construction phase; and
• Deployment set is the focus of the transition phase.
• The other set i.e. the *'Management set'* takes a check on the balance roles. This set also evolves across the life cycle but at a fairly constant level.

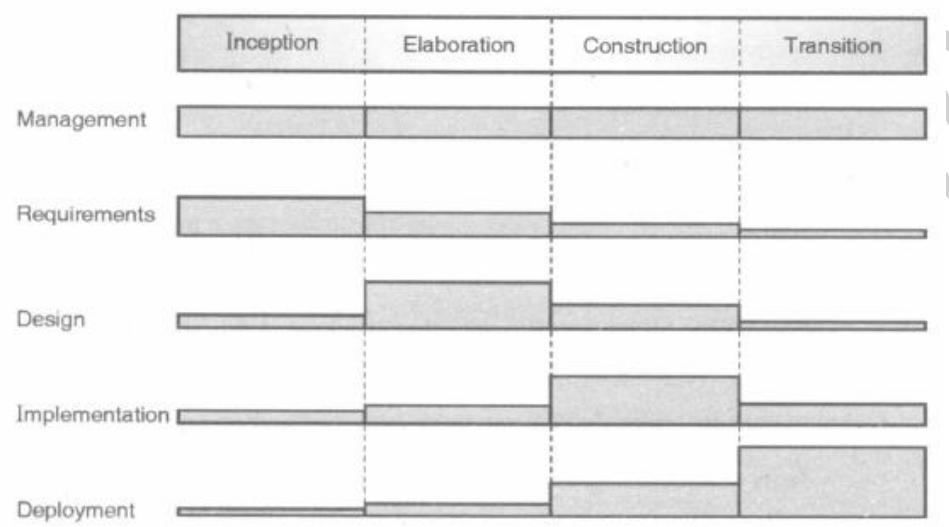


Fig. 5.1.2
Mostly, almost all the software development tools map closely to one of the five artifact sets.
2. **Management set** : scheduling, workflow, error tracking, version control, documenting, deriving spreadsheets, resource management and presentation tools
2. **Requirements set** : requirements management tools
3. **Design set** : design modeling tools
4. **Implementation set** : compilers, debuggers, code analyzers, test coverage analysis
tools, and test management tools
5. **Deployment set** : test coverage analysis and test automation tools, network management tools, commercial components (like the OS, GUI, RDBMS, Networks, Middleware), and the installation tools.

Advantages of Deployment Set over other sets (Deployment set v/s other sets):

This differentiation is important because the structure of information delivered to the user is very different from the structure of the source *r* code information.
• The implementation set consists of the source code whereas the deployment set consists of executable code.

• The quality achieved in the deployment set is really not that attained in the design and implementation sets because the deployment set consists of :
o Dynamically reconfigurable parameters includes any type of run-time parameters, buffer sizes, color palettes, number of servers, number of simultaneous clients, and database files.
o Impact of compiler and linker optimizations includes space optimization versus speed optimization.
o Performance issues in case of any type of allocation strategies such as centralized v/s distributed, primary v/s shadow threads, dynamic load balancing, hot backup v/s checkpoint/rollback)
o Virtual machine constraints such as file descriptors, garbage collection, heap size, maximum record size, and disk file rotations,
o Issues in Process-level concurrency such as the deadlock and race conditions,
o Impact of varied Operating systems or any other platform differences on the performance or behavior of the software product.

5.1.3 Artifact Evolution over the Life Cycle Phases

Each phase of development life cycle focuses on a particular artifact set. But at the end of each phase, the overall system state will have covered all the artifact sets as shown in the below Fig. 5.1.3 :

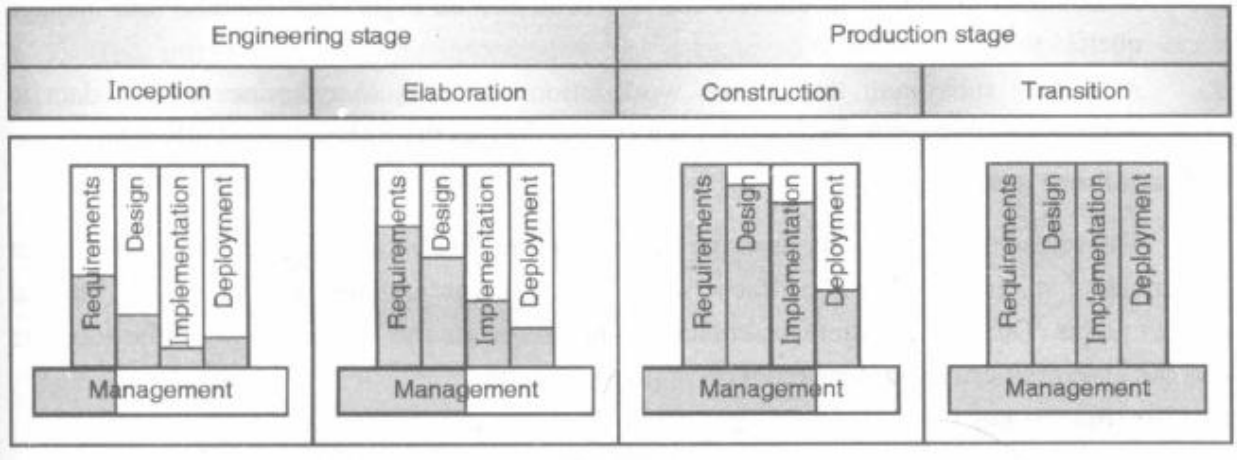


Fig. 5.1.3

• The **inception** phase mostly focuses on the user's requirements with a secondary preference on initial deployment view.
• The **elaboration phase** goes in greater depth in requirements, much more depth in the design set, and further works on implementation and deployment issues.
• The **construction** phase mostly focuses on design and implementation.
• The **transition** phase focuses on achieving the consistency and completeness of the deployment set in context of other sets.

5.1.4 Test Artifacts

• The *test artifacts* are developed simultaneously along with the product since from the inception phase through the deployment phase. That means, testing is a complete-lifecycle activity that is conducted in each and every phase of the life cycle. That means, testing is started 'early' in the development life cycle and it is NOT a late life-cycle activity'.
• The test artifacts are communicated and developed within the same artifact sets as that of the developed product.
• The test artifacts are implemented in programmable and repeatable formats.
• The test artifacts are documented in a similar manner as any software product is documented.
• The test artifact developers/engineers use same tools, techniques, and training as that of the software developers/engineers use while developing the product.
• 'Test artifact sets' are very *project-specific.*

Example :

Consider a software system which performs seismic data processing for the purpose of oil exploration. This project has three fundamental subcomponents:
1. A sensor to capture raw seismic data in real time
2. A technical operation to convert the raw data into an organized database and manage queries to this database
3. A display subsystem that allows workstation operators to examine seismic data in human-readable form. Such a software system derives the following test artifacts:

Management set:

The release specifications describe the objectives, evaluation criteria, and results of an intermediate milestone. These artifacts work as test plans and test results among internal project teams. The changes (defects, enhancements, requirements' ambiguities) in the software and the entry/exit criteria associated with it are again tested.

Requirements set:

The use cases associated with the system involves the operational concepts of the system, acceptance test cases, expected functionalities of the system and its quality attributes. The requirement set is also called as 'Test Artifact' because it is the base for all the assessment activities throughout the life cycle.

Design set:

It includes the test models for non-deliverable components which are used to test the product. These components include design set artifacts that are a seismic event simulation for attaining realistic sensor data, a 'virtual operator' that will support the unattended and afterhours test cases. It also includes the instrumentation suites needed for early demonstration of resource usage, the response time, test case drivers for the whole system and for the standalone components.

Implementation set:

This artifact set includes the source code representations useful for testing the components and also the test drivers provide the equivalent test procedures and test scripts. This source code may also include human-readable data files representing explicit test source files. The output files of these test drivers provide the equivalent test reports.

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

Deployment set :
This artifact set provides the executable versions of test components, test drivers, and datafiles.

5.2 Management Artifacts

The management set consists of various artifacts that describe the intermediate results and the associated information that is required:

- To document the product/process legacy,
- To maintain the product,
- To improve the product, and
- To improve the process.

Business Case Artifact:

- This artifact gives all the information that is required to decide whether the project is worth investing in or not.
- It describes the expected cost, expected technical and management plans, and backup data necessary to face the risks and realism of the plans.
- The main objective is to transform the vision into economic terms which can help the organization to make an accurate assessment of ROI (Return on Investment).
- The financial forecasts are updated step by step with more accurate forecasts as the life cycle progresses.

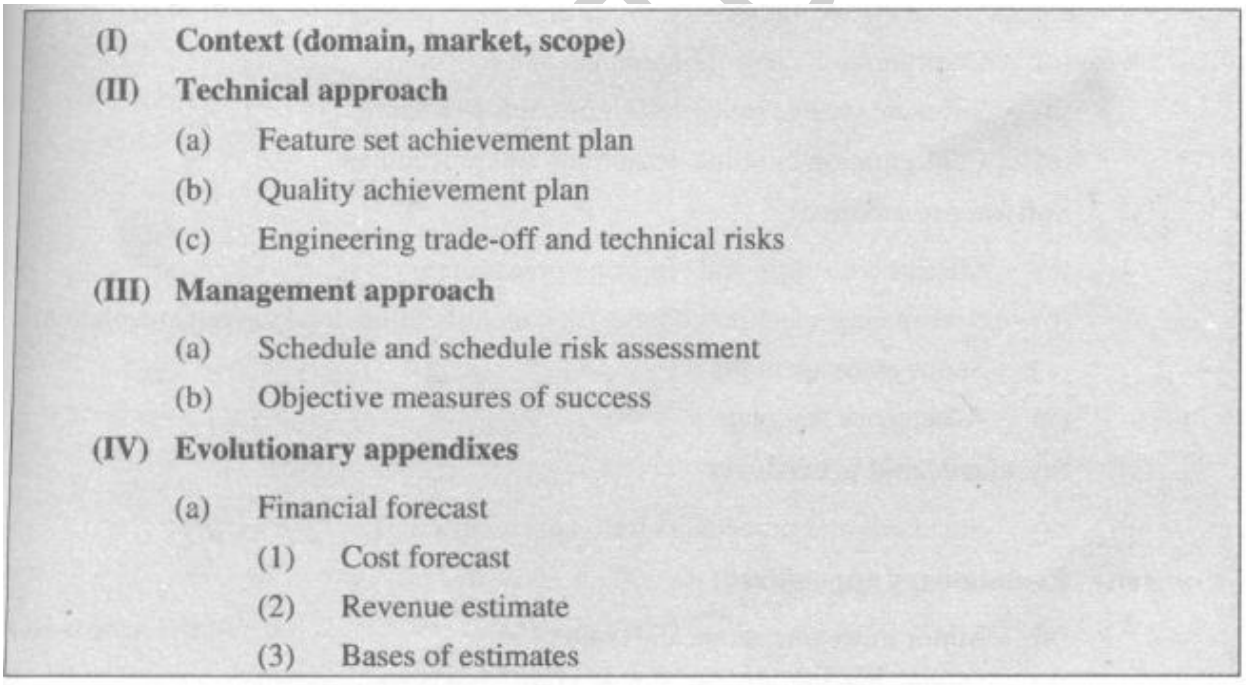


Fig. 5.2.1: Outline of a Business Case

Software Development Plan Artifact:

The software development plan (SDP) gives in detail description of the process

framework. Two main objectives of a SDP are:

1. Periodic updating and

2. Understanding and acceptance by managers and practitioners.

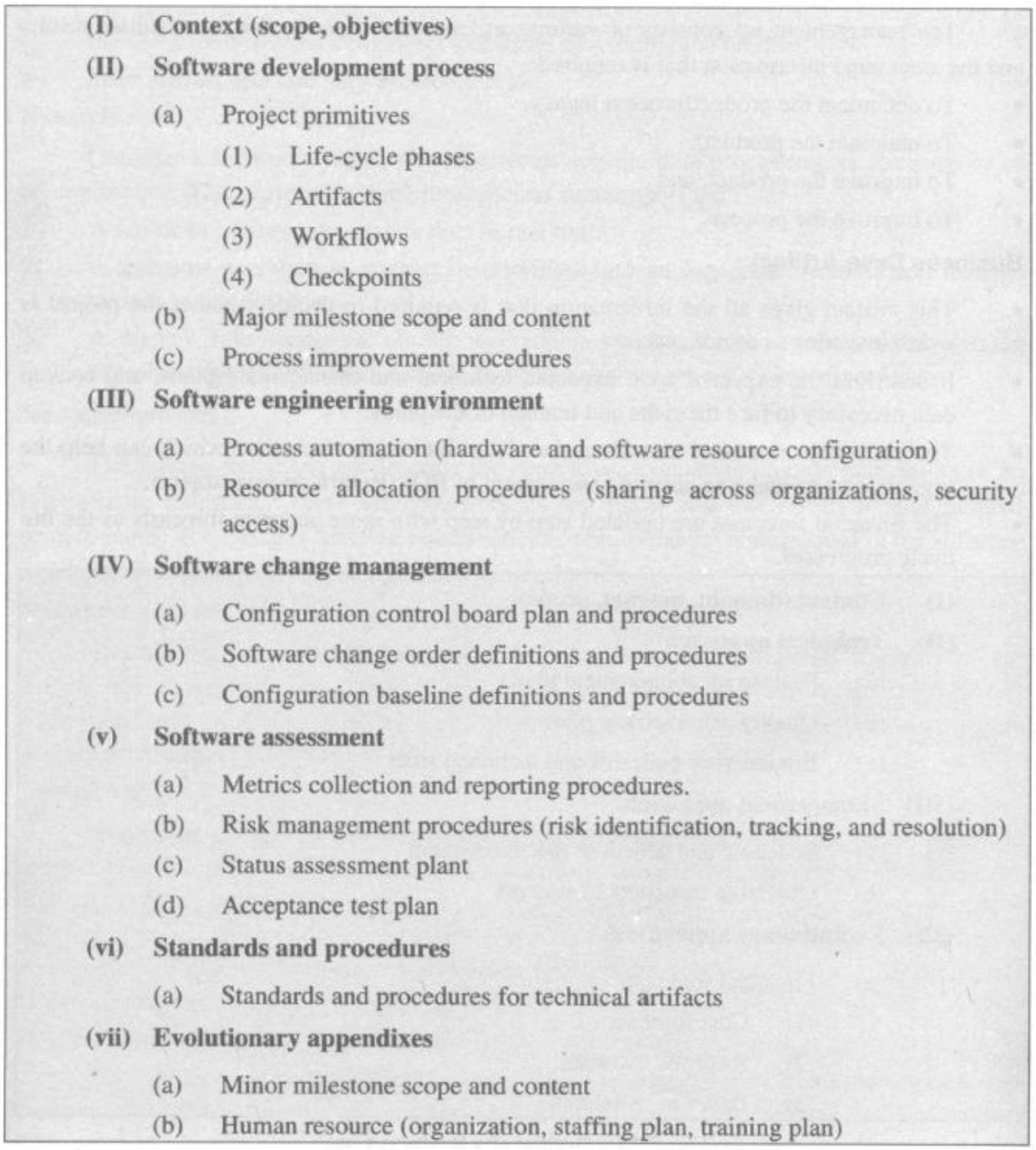


Fig. 5.2.2 : Outline of a Software Development Plan

Work Breakdown Structure :

Work breakdown structure (WBS) is a base for budgeting and cost estimation used to monitor and control the project's financial performance. The project manager must have an insight into project costs and expenditure. The WBS is a serious project planning constraint.

Software Change Order Database :

• Managing and tracking the changes is one of the important activities in an iterative development process.

• As the iterative development process provides a 'greater change freedom', a project can be iterated again and again to achieve more and more productivity.

• This flexibility of making greater changes in the project ultimately increases the number of iterations and thus increases the content and also the quality of the software product and that too within a given schedule.

• The flexibility of making changes is achieved in practice through automation and using the today's iterative development environments which involve a phase for 'change management'. Because without automation, the organizational processes that depend on manual change management techniques face lot of major inefficiencies.

Release Specifications :

• This artifact includes the scope, plan, and objective evaluation criteria for each release and all these details are derived from the vision statement as well as from many other sources such as from the analysis, risk management concerns, architectural considerations, implementation constraints, and quality thresholds.

• All these artifacts of the Release specification evolve and get updated along with the process, thus, achieving greater fidelity and maturity in understanding the requirements.

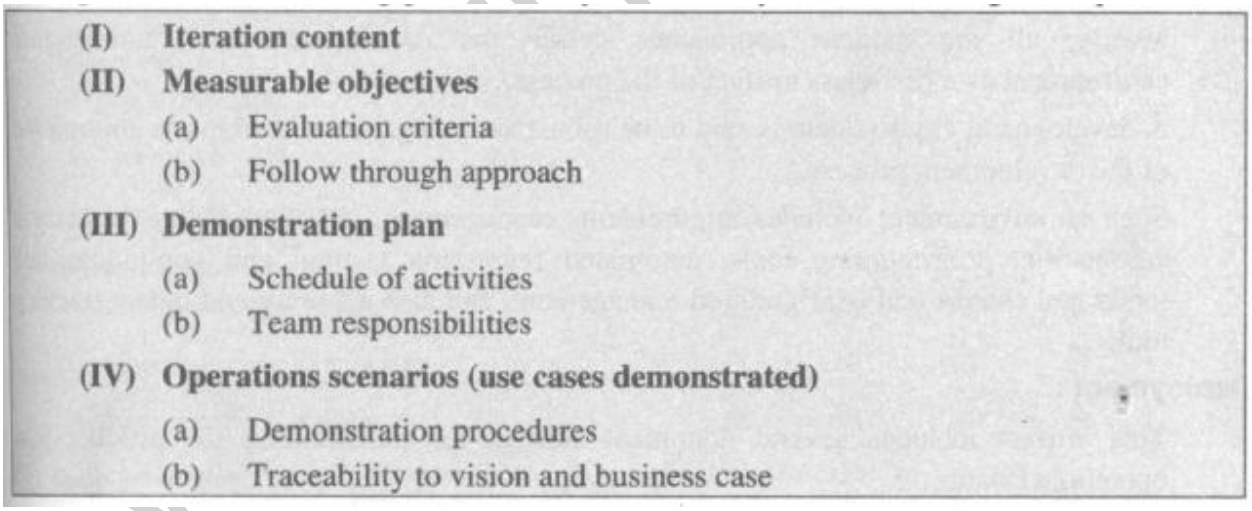


Fig. 5.2.3 : Outline of a Release Specification

Release Descriptions :

• It describes the results of each release that includes the performance details against each of the evaluation criteria in the corresponding release specification.

• Release descriptions also describe the evaluation criteria for the configuration baseline and also provide substantiation i.e. a thorough demonstration, testing, inspection, and analysis of each criterion that has been addressed as required.

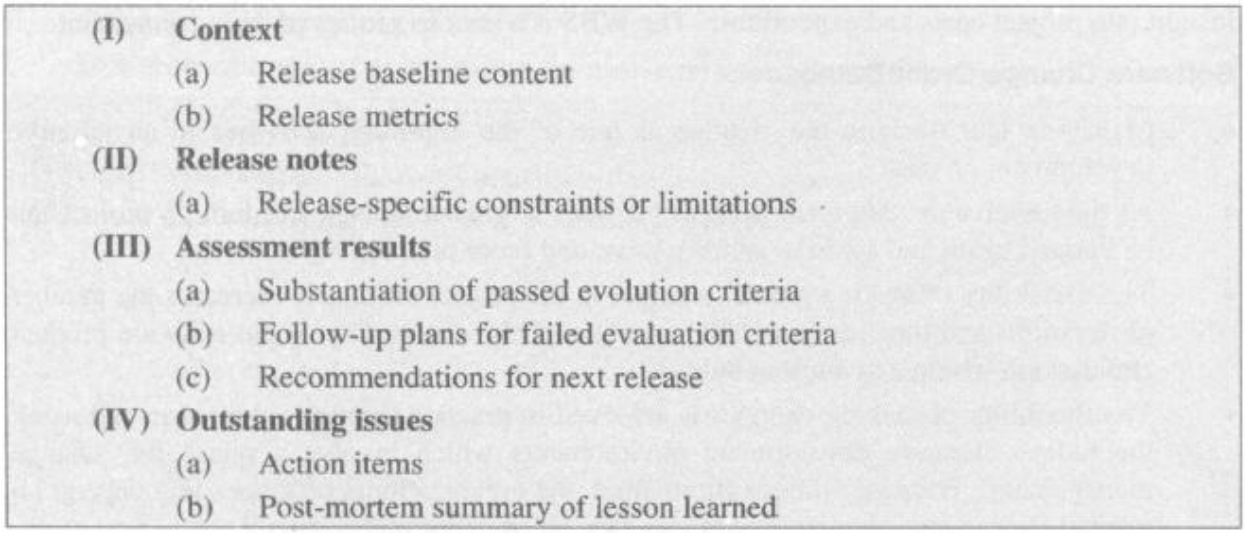


Fig. 5.2.4 : Outline of a Release Description

Status Assessments :
• These artifacts provide periodic snapshots of the project status that includes the project manager's risk assessment, and the quality and management indicators.
• It also includes a review of resources, staffing, financial data (cost and revenue), top ten risks, technical progress, major milestone plans and results, and scope of the project.

Environment :
• Mostly, all the modem approaches define the development and maintenance environment as a first-class artifact of the process.
• A development environment is said to be robust and integrated if it supports automation of the development process.
• Such an environment includes requirements management, visual modeling, document automation, programming tools, automated regression testing, and continuous and integrated change and configuration management, and also a feature and defect tracking tool.

Deployment:
• This artifact includes several document subsets for transitioning the product into operational status.
• If the system is delivered to a separate maintenance organization, then in such cases, the deployment artifacts must include the computer system operations' manuals, software installation manuals, plans and procedures, site surveys, and etc.
• If the software pioduct is used for commercial purpose, the deployment artifacts must include marketing plans, sales rollout kits, and training courses.

Management of Artifact Sequences :
Each phase of the life cycle produces new artifacts and also updates the previously developed artifacts to incorporate the changes done and describes the further depth and breadth of the solution.

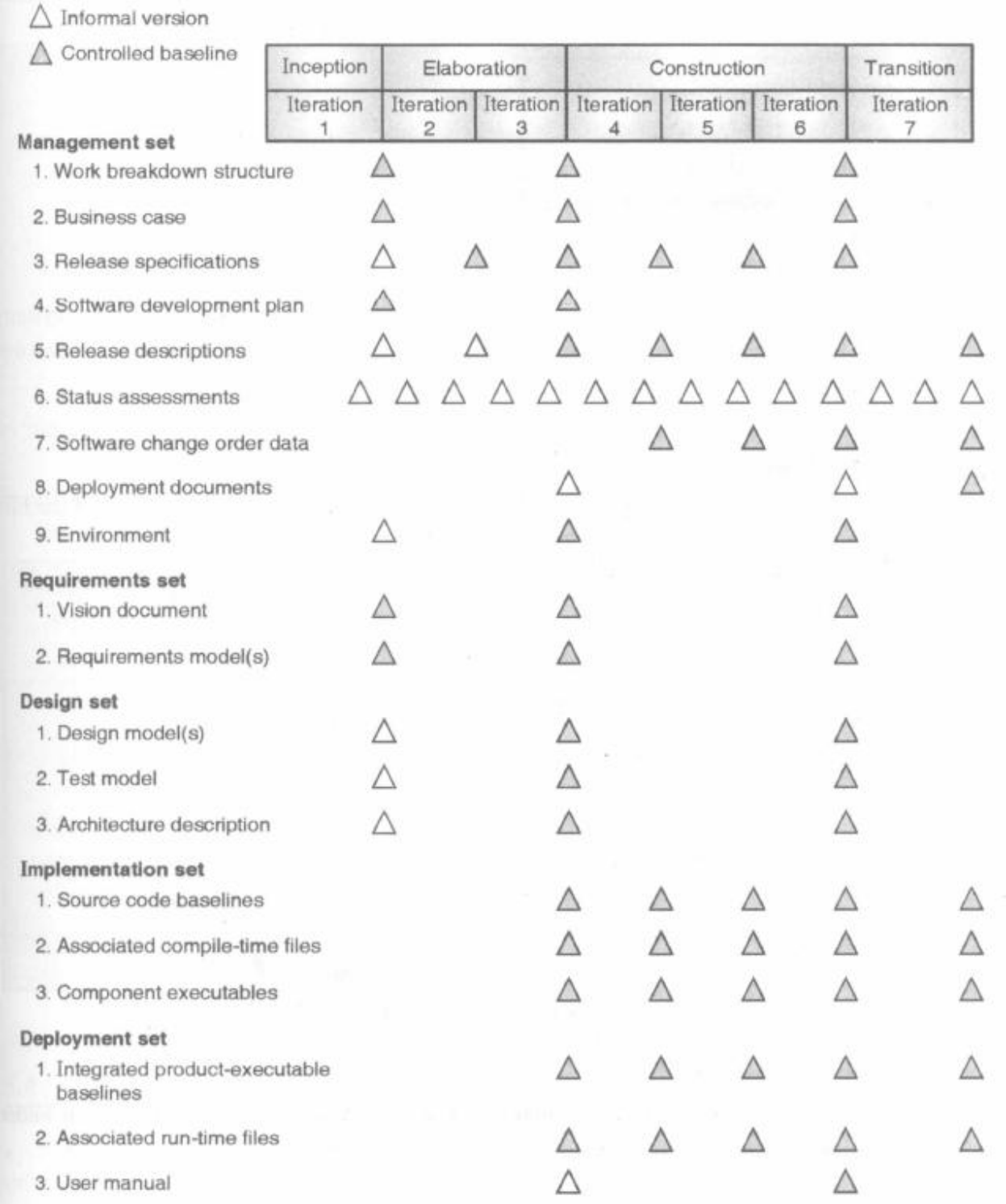


Fig. 5.2.5 : Artifact Sequences across a typical life cycle

5.3 Engineering Artifacts

This set of artifacts is derived from the thorough engineering notations such as of UML notations, programming languages, or executable machine codes. This Engineering artifact set can be categorized into three kinds which are specially intended to give a more general review. These are :

1. Vision document
2. Architectural description
3. Software user manual

1. Vision Document:

• This document provides a complete vision about the proposed software system that is under development and. It is the base that supports the contract between the *funding authority* and the *development organization.*

• This document involves changes progressively as understanding evolves out of the requirements, architecture, plans, and technology. A good vision document should change slowly at each and every phase of the life cycle.

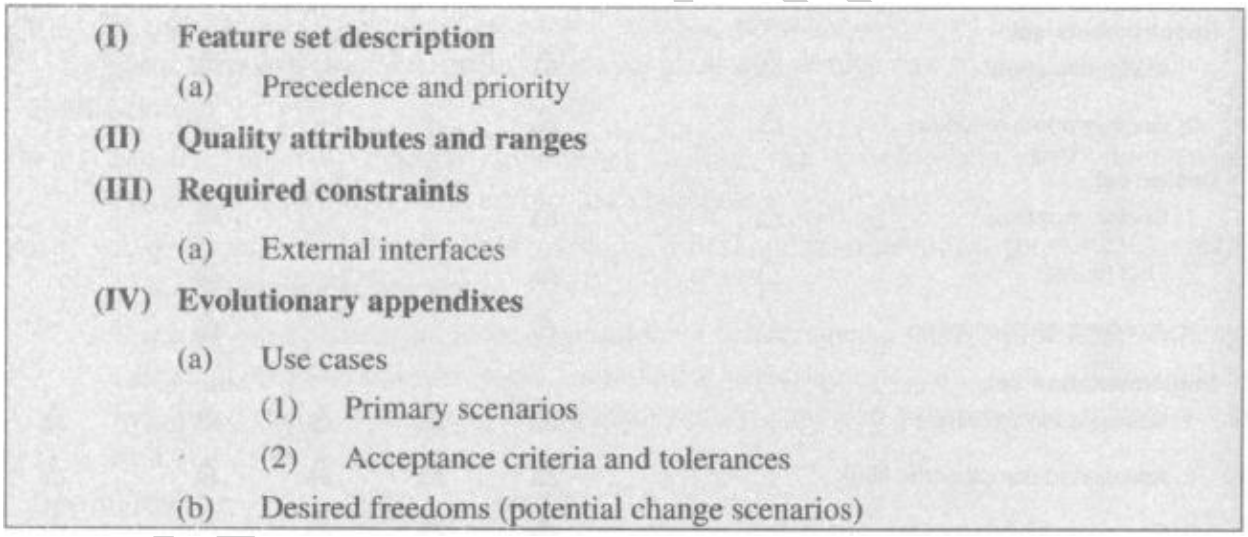


Fig. 5.3.1 : Outline of a vision Document

2. Architecture Description

• This artifact provides an organized architectural view of the software that is under development.

• These descriptions are derived from the design model which includes the views of the design, implementation, and deployment sets that helps in understanding how the operational concept of the requirements set are derived.

• The depth and breadth of the architecture description varies from project to project based upon various factors.

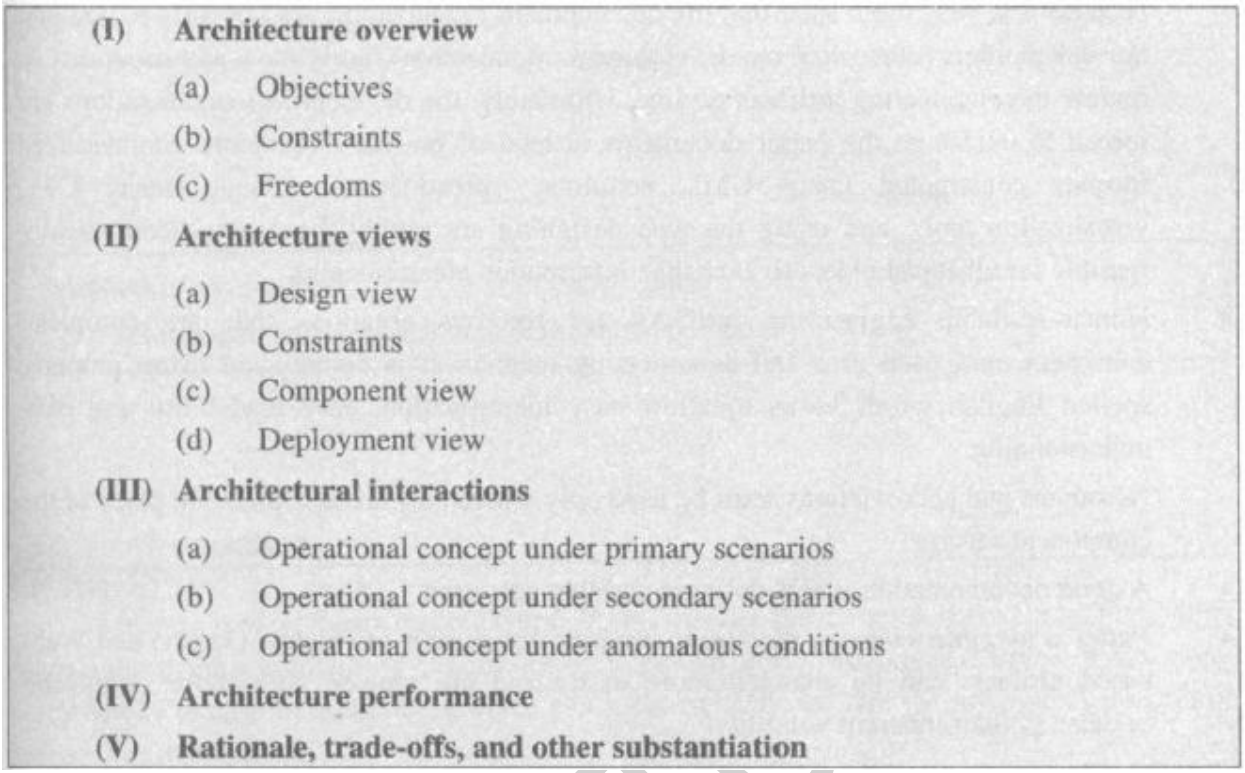


Fig. 5.3.2 : Outline of a Architecture Description

3. Software User Manual

• The manual provides the user with the references that are required to support the delivered software.

• The user manual includes the installation procedures, usage guidance, operational constraints, and a user interface details.

• This manual must be developed in early phases of the life cycle since it is a required mechanism for communicating and stabilizing the user requirements.

• The user manual is written by the members of the test team who understand the user's perspective more clearly than the development team docs.

5.4 Pragmatic Artifacts

• *People are interested in reviewing the Artifacts but they don't understand the language o f the artifact* because they do not have any knowledge of the engineering language in which the artifact is written and they don't even care to learn it. It is very common among the software managers, quality assurance specialists, or an auditing authority who say "I do know much about the UML notations and nor I'm going to learn UML, but I can review the design models of this software, so give me the software description in the form of flowcharts or text so that I can understand."

• *People are interested in reviewing the artifacts but don't have any access to the tools* because it is very often seen that the development organization are not fully tooled and the stakeholders

(other than the development organization) rarely have any capability to review the engineering artifacts on-line. Ultimately, the development organizations are forced to exchange the paper documents instead of on-line documents. Standardized formats constructed using UML notations, spread-sheets, Visual Basic, C++, visualization tools, and using the web designing are rapidly becoming economically feasible for all stakeholders to exchange information electronically.

• Human-readable engineering artifacts use rigorous notations that are complete, consistent, and used in a self-documenting manner. It is constructed using properly spelled English words so as to allow easy identification, easy readability and easy understanding. Acronyms and abbreviations must be used only where they are acceptable in place of the component's usage.

• A good documentation is self-defining and that gets used.

• *Paper is tangible whereas, electronic artifacts are so easy to change.* On-line and Web-based artifacts can be changed more easily and are viewed with more skepticism because of their inherent volatility.

Review Questions

Q . 1 Give an overview of the artifact set.
Q . 2 Write short notes on :
(a) Management artifacts
(b) Engineering artifacts
(c) Pragmatic artifacts
Q. 3 Write short notes on Test Artifacts
Q. 4 List out the different types of management artifacts and give their brief outline.
Q. 5 List out the different types of engineering artifacts and give their brief outline.

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

6. Model Based Software Architecture

Syllabus :
A Management perspective and Technical perspective.

6.1 Introduction

Definition of Software Architecture: If we consider the complex software system which shows the overall problem by central design is known as *Software architecture.* It has many dimensions in case of complexities. We can measure or prove that by any logical relation of physics or mathematics.
It is note that without depends on any prove theory software architecture based on the experimentation. We can say this is fundamental reason of the transitioning to an iterative process. Old conventional software process produces the less powerful systems. Architecture comes First are includes
• Simpler
• Requires informal representations
• Requires single computers
• Requires single program system
• Various mapped objects
• Objects implementation
Model: Independent abstraction of a system is known as **model.**
View : Model which abstract specific, relevant perspective is known as **view.**

6.2 Architecture: Management Perspective

Architecture is most critical and technical product of the software project, which includes parameters infrastructure, control and data interface. All these parameters coordinates each other and makes huge system. If there are lots of languages available in the communication media and management team members having different literacy then lots of communication problems are generated which are never solved. If the team members have good in inter project communication then and then software architecture must be accurate and exact.
There are three different aspect of Management Perspective;

1. Architecture
• It is design of the software system
• Intangible concept
• All types of engineering which is useful to fulfill bill of material is part of it.
• Problems of whatever purchase and sale of the components are solved since custom components are elaborate. Cost of unit component and it's assembling and construction cost is to be determined.
2. Baseline of Architecture
• It is tangible artifacts

• They are satisfies the all stakeholders
• All stakeholders are the primary vision which includes the function and quality
• The vision set the business parameters which includes
o Cost
o Profits
o Time
o Technology
o Peoples
3. Description of Architecture
• Well manner subset of information
• It includes the text and graphics which gives the complete information about the model
• It communicates about the intangible information for tangible artifacts.
If system changes then the architecture should be change. The number of views and its level should be change. For example of raw boat and jet boat, both are boats but is run through human interface and another is automatic.

Fig. 6.2.1: Architecture of raw boat and jet boat

Importance of Software Architecture
1. Get stable software architecture, which is fixes the milestone. By using those milestones problems like purchasing and selling are resolved.
2. Gets basis of the trades off balancing between problem and its solution space.
3. Encapsulates the communications between individual, various teams, stakeholders and organizations.
4. Project unsuccessful or failure reasons : Dull Architecture and immature process
5. Requirements are understandable in the mature process and architecture should be previously demonstrated.
6. Two important facts 1) Architecture Development and 2) Process Definition

6.3 Architecture: Technical Perspective
We look towards Software via various different aspects which are stated below;
• Structure of software system
• Behavior
• Guides lines patterns about elements
• Collaboration
• Compositions

The all over information given in design model and architecture view is abstraction of design model. Real system involves the four views of the design model;

Table 6.3.1: Design model's four view

Design	Express structure and function of the design model. Important for every system
Process	Express concurrency and control thread between other views Add as per complexity
Components	Express implementation set structure Add as per complexity
Deployment	Express structure of deployment set Add as per complexity

Following figure illustrates artifacts of the design set which includes above various views and architecture description (captures electronically and maintains on printed format). Engineering models and views are defined with collection of the Unified Modelling Language diagrams such as use case diagram which describes how system critical.

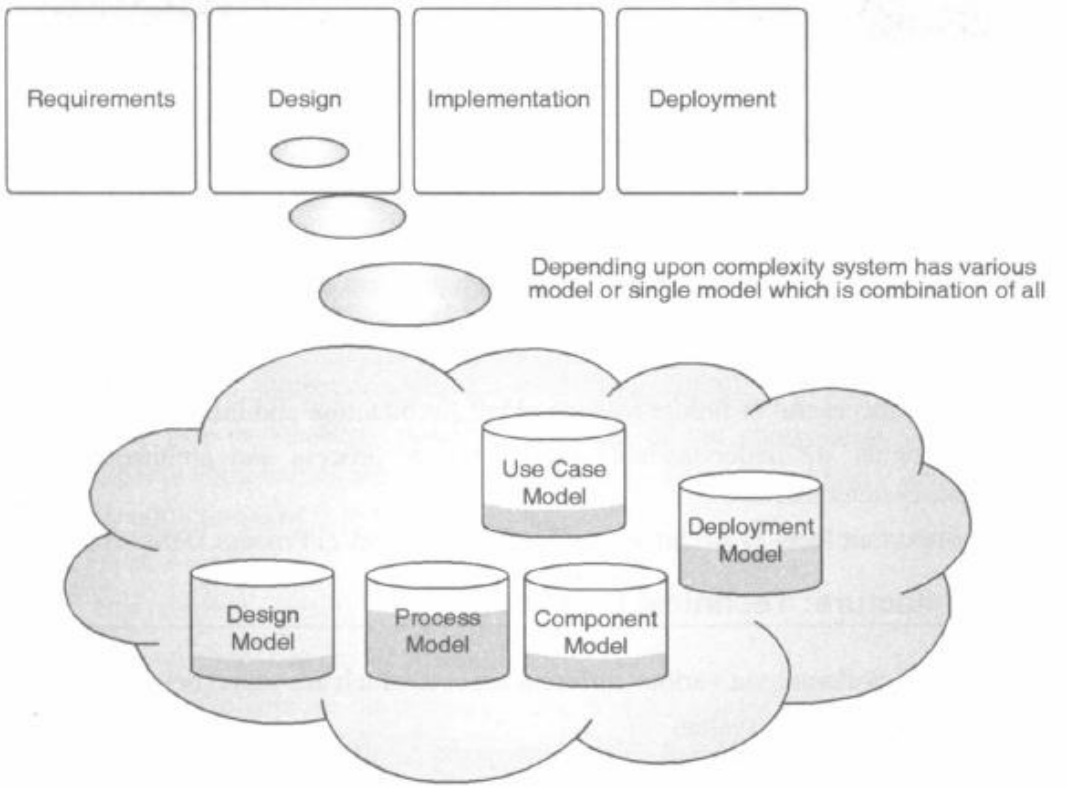


Fig. 6.3.1 : Architecture, organized and abstracted view into design model

Review Questions

Q. 1 What is Software architecture ?
Q. 2 Explain three aspects of management perspective.
Q. 3 Why software architecture is important ?
Q. 4 Explain Software architecture in technical perspective.

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

UNIT – III

7.Work Flows of the Process

Syllabus:
Work flows of the process : Software process workflows, Iteration workflows.

7.1 Old Way of Process Workflow - Sequential Activities

It is a common assumption that the requirements are stable in all these models and the chances of returning back to the earlier phases is very less. That means, the human nature 'keep It Simple Sweet (KISS)' mostly assumes that all these process models perform activities sequentially.

Drawbacks:

• Projects 'progressed' through fixed sequential activities i.e. the other activity would be conducted only after completing the previous activity.
• Successful projects did not have any rigid boundaries.
• Unsuccessful projects did have rigid boundaries.
• Much effort was spent on defining the details i.e. completing the manual and etc; while slight importance was given for development and engineering activities.

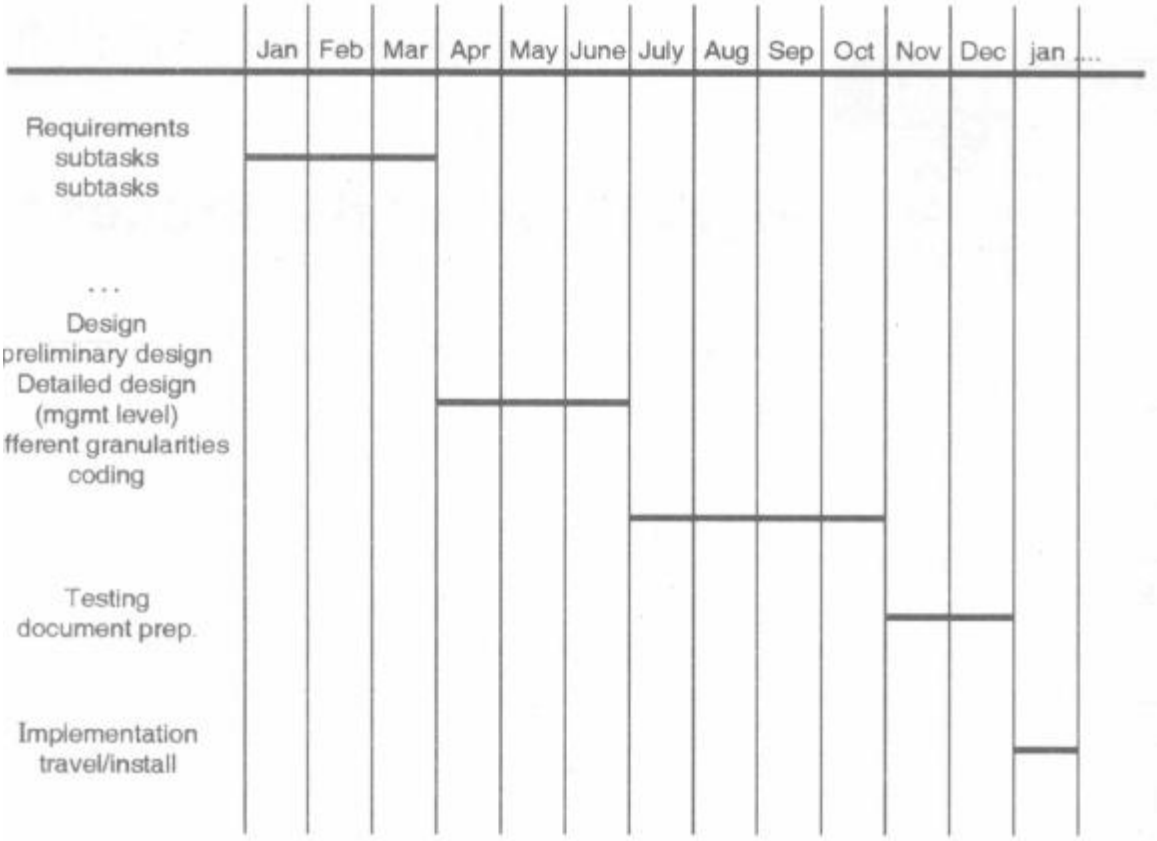


Fig. 7.1.1 : Gantt chart

7.2 New Way of Process Workflow - State of the Project

In the new way of software development, it is completely opposite - developing a complex software requires multiple teams to work in parallel on major functional areas in a synchronized manner, and then these functional areas are evaluated separately and then at last they are integrated with each other to form a complete system. In this process, each individual team works on their own life cycle phases planned for their module. And this is how we can do more in less time utilizing more tools with better support environments.

Advantages :

• Naming the life-cycle phases after the predominant activities is *avoided.*

• Focus is on tracking the *state* of development within phases (inception, elaboration, construction, transition) via the minor milestones (end of each iteration).

• Development is done in increments progressing along way.

7.3 Software Process Workflows

We have already learnt in the 'Software Engineering' subject about the various types of models such as the linear sequential (waterfall), incremental , concurrent, RAD, spiral, JAD that can be used for software development.

All these models have five life cycle phases in common :

1. Analysis
2. Design
3. Coding
4. Testing
5. Deployment

Barry Boehm describes this 'unified approach' of software development in seven major ***'Work Flows':***

1. Management
2. Development Infrastructure
3. Requirements
4. Design
5. Implementation
6. Evaluation
7. Deployment

Boehm's 7 major Workflows :

1. Management :

• This workflow involves planning and controlling of the process so as to achieve product and project goals.

• The purpose of software project management is balancing the competing objectives, managing risk, and overcoming constraints so as to deliver a successful product which meets the needs of both customers and users.

• Active workers during this process : Project manager and Project Reviewer.

• Works of Project Manager

o Initiate project

o Develop iteration plan

o Develop quality assurance plan

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

o Monitor project status
o Schedule and assign work
o Report status
o Handle exceptions and problems
o Works of Project Reviewer :
o Project approval review
o Project planning review
o Iteration plan review
o Iteration evaluation criteria review

2. Development Infrastructure :
• This defines the infrastructural requirements, installation platform and introduction of *change management.*
• It provides the software development organization with the software development environment -both processes and tools - that are essential for the development team.

• **Active workers during this process** : Process Engineer, Software Architect and Tool Specialist.
• Works of Process Engineer :
o Development case
o Project specific templates
• Works of Software Architect:
o Design guidelines
o Programming guidelines
• Works of Tool Specialist:
o Tool guidelines
o Tools

Configuration and Change Management:
• The purpose of Configuration and Change Management (CM) is to monitor and administrate changes in the project work so that they are consistent with the requirement.
• The goal of CM is to release and control the version (changes made) of the system.
• Active workers during this process : Configuration Manager and Change Manager.
• Works of Configuration Manager :
o Set up CM environment
o Establish CM policies
o Write CM plan
• Works of Change Manager :
o Establish change control process
o Review change request

3. Requirements (& Analysis):
• It involves ***analysis*** of the problem and determining scope of the project.
• The purpose of this workflow is to elicit the requirements for the project, including their identification, modelling, and documentation.

• The goal of this process is the Software Requirements Specification (SRS) which encompasses the captured requirements. This describes what the system should do and allows the developers and the customer to agree on that description.
• Goal of Analysis : to determine risks, stability of the product and expenses of resources.
• Active workers during this process : System analyst and software architect.
• Works of system Analyst:
o Elicit stakeholders requests
o Develops requirements management plan
o Finds actors and use-cases
• Works of software Architect:
o Prioritize use-cases
o Review requirements

4. Design :
• The purpose of this workflow is to ***develop a robust architecture*** for the system based on the requirements, to transform the requirements into a design, and to ensure that implementation issues are reflected in the design.
• The goal of Design is to show *how* the system will be understood in the implementation phase.
• *Active workers during this process* : Architect and Designer
• *Works of Designer :*
o Use-case analysis
o Use-case design
o Class design

5. Implementation:
• The purpose of implementation is to ***code and test*** the system.
• The goal of implementation is to develop ready to execute module independent of other modules.
• Active workers during this process : Implemented integrator and code reviewer.
Works of Implementer:
o Implement a module
o Fix a defect
o Perform unit testing
Works of Integrator:
o Plan system integration
o Plan subsystem integration
o Integrate subsystem
o Integrate system
Work of Code Reviewer :
Review code
6. Evaluation :
• The purpose of this workflow is to design test case procedures and other verification methods.
• The goal of test process is to design and execute test cases for the system in order to eliminate defects.
• Active workers during this process : Test designer, tester

Works of Test Designer :
o Plan test
o Design test
o Implement test
o Evaluate test

Work of Tester :
Execute test
• It also involves assessing different trends in cost, time, quality, deliverables and communications.

7. Deployment:
• The purpose of deploy is to install the software at the end-user.
• The goal of deployment is to deliver the system to the user community. It involves issues of marketing, packaging, installing, configuring, supporting the user-community, making corrections, etc.
• Active workers during this process : Deployment manager and technical writer.
• Works of Deployment Manager:
o Develop deployment plan
o Manage acceptance test
o Define bill of materials
Works of Technical Writer:
o Write release notes
o Develop support materials

7.4 Workflows and Life Cycle Phases____

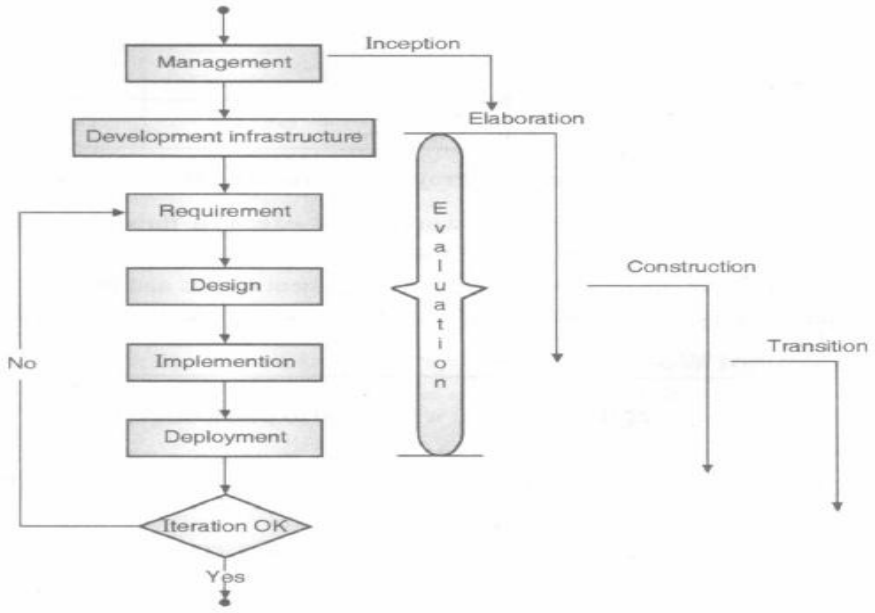


Fig. 7.4.1 : The Four Life Cycle Phases and the Seven Workflows

The above figure illustrates the *relationship o f workflows with the life cycle phases.*

You can also see that these workflows are executed in an iterative manner where the following workflows are iterated in each phase of the life cycle: Analysis - Design - Implementation - Evaluation.

In 7.4.1 figure illustrates the relationship between workflows and life cycle phases and the management effort and intensity in each workflow.

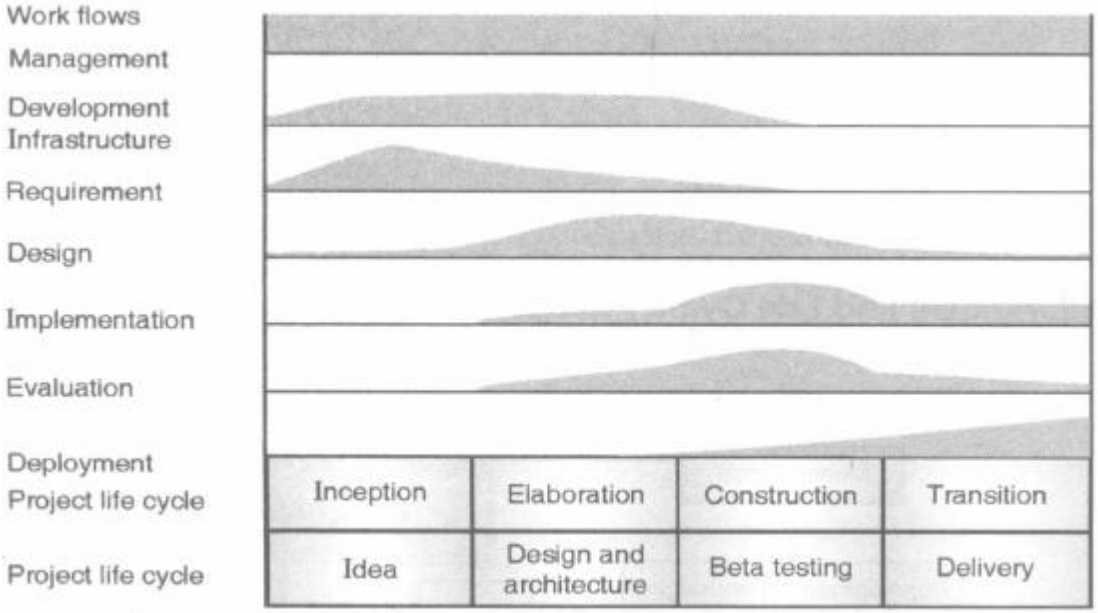


Fig. 7.4.2 : Workflows related to Project and Product Life Cycle Phases

You can see that the 'Management' workflow is executed throughout the life cycle period as it manages the umbrella activities of the Software Process. It is also seen that in other workflows, the management efforts and intensity increases as we forward from 'development infrastructure' workflow towards 'deployment' workflow.

7.5 Management Workflow

There are various reasons (listed below) which lead to project failure:

- Badly defined system requirements
- Poor reporting of the project's status
- Unmanaged risks
- Poor communication among customers, developers, and users
- Use of immature technology
- Inability to handle the project's complexity
- Sloppy development practices
- Unrealistic expectations
- Inability to move beyond individual and personality conflicts
- Stakeholders' Politics

To avoid these above failures, there is a need of proper *Project Management.*

Management is all about:

- Creating an environment suitable to develop critical projects.

• The activity involved in ensuring that software is delivered on time and on schedule, in budget and in accordance with the requirements of the organisations developing and procuring the software.

• Management workflow *oversees all the life cycle phases* and *supports further workflows.*

Table 7.5.1 : Activities and Outputs of the life cycle phases in 'Management workflow'

Life Cycle Phases	Activities	Outputs
Inception	• Analyze requirements, • Prepare business proposal, • Plan project • Launch project	• Scope • Statement of work (SOW) • Requirements Definition and Description (RDD) • Software requirement Specification (SRS) • Customer acceptance to project plan • Win-Win contract • Project deliverables
Elaboration	• Modeling • Designing and architecture • More specific development of project plan • Organisation of resources • Prepare work breakdown structure (WBS)	• Designing and architecture models • Development plan • Project organisation • WBS
Construction	• Implementation of project plan • Coding – design and code • Testing – reviews and walkthroughs, system testing, alpha and beta testing. • Integration of components • Monitor and control - time, cost, quality and scope	• Project schedule • Progressive status • Tested products • High degree of customer acceptance • A quality software product ready to move to the deployment platform

Life Cycle Phases	Activities	Outputs
Transition	• Conversion of tested product into packaged executables • Deploying product at customer environment • Training of the customers • Change management plan – creating a timeline for switching over to the new environment	• Packaged executables • Plan for customer preparedness • acceptance of the product • User preparedness • Project closure and sign off

Ultimate output of **Management workflow** includes following **Artifacts :**
• Business Case with explanations
• Software Development Plan (SDP)
• Progressive Status Assessment
• RDD, SRS, SOW and WBS

7.6 Development Infrastructure Workflow

Objective : to create a development environment in-house and at the customer's site to develop and then to deploy the s/w.
This environment (infrastructure) involves major technical issues, major functional specifications and features that must be delivered to the customer.

Table 7.6.1 : Activities and Outputs of the life cycle phases in 'Development Infrastructure workflow'

Life Cycle Phases	Activities	Outputs
Inception	• SRS analysis • Find alternative development strategies • Choose the best and cost effective strategy • Determine the development platform	Development platform in terms of : • h/w and s/w, • technologies and • tools defined for implementation.

Life Cycle Phases	Activities	Outputs
Elaboration	• install the development platform using internal resources • meet the inadequacies and non-availability through procurement • design and develop database for software development and testing	• installation of development platform • Test database
Construction	• maintain development environment • maintain software change order database	• ready to serve the s/w development needs
Transition	• to maintain environment and software change order database	Keep ready the changed database and deployment the required environment to receive beta tested s/w product.

Ultimate output of **Development Infrastructure workflow** includes following Artifacts :
• A defined environment such as tools, techniques, methods, procedures, standards, version / configuration control, etc.
• Software Change Order database

7.7 Requirements Workflow :

The Requirements workflow is the core part of the project life cycle as well as the product life cycle. The following workflows operate on the output of this workflow.
Objective : Convert the requirements into operational modules and define their sequence of development and schedule of release.

Table 7.7.1 : Activities and Outputs of the life cycle phases in 'Requirements workflow'

Life Cycle Phases	Activities	Outputs
Inception	• Analyze requirements • Convert requirements into operational modules • Arrange these modules in the order of development and release schedule.	• Defined operational modules • Logical sequence of development • Schedule of release

Life Cycle Phases	Activities	Outputs
Elaboration	• Define architecture objectives based upon the modules defined in inception phase and based upon the expectations set by the customer.	• Definition of architecture objectives to be designed accordingly.
Construction	• Study and evaluate the different architecture options and select one of them that maximize the level of objective achievement.	• Definition of objectives and achievements to be achieved in the construction phase.
Transition	• Analyze the release plan • Revisit and cross-check the release objectives	• Achievement of release objectives

Ultimate output of **Requirements workflow** includes following **Artifacts :**
• Requirements set - business case, risks, and operational modules.
• Release Specifications - sequence of development and schedule of release, training, customer support.

7.8 Design

Objective : To obtain effective design and its architecture with the baseline and components built accordingly.

Table 7.8.1 : Activities and Outputs of the life cycle phases in 'Design workflow'

Life Cycle Phases	Activities	Outputs
Inception	• Study different architectural concepts and formulate the best suitable one.	• Logical and physical model of the architecture
Elaboration	• Achieve architecture baseline using the conceptual model	• Baseline architecture
Construction	• Design components of the achieved architecture	• Architecture design
Transition	• Refine architecture and components so that it can be moved onto customer deployment environment	• Architecture design for deployment

Ultimate output of **Design workflow** includes following **Artifacts :**
• Design Models
• Architectural Models

7.9 Implementation & Evaluation

Objective : Assessing the software for fulfilment of scope, deliverables and quality.

Table 7.9.1 : Activities and Outputs of the life cycle phases in 'Implementation & Evaluation Workflow'

Life Cycle Phases	Activities	Outputs
Inception	• build prototype • test prototype • evaluate	• refined design and architecture
Elaboration	• Design and develop software system	• software product
Construction	• Release software for alpha and beta testing	• Improved software product • Implementation on deployment platform
Transition	• Conduct customer acceptance testing	• Customer acceptance

Ultimate output of **Implementation workflow** includes following **Artifacts** :
• Architectural (physical and logical) models
• Components
• Packages
• Programs
• modules
• software to be released
Ultimate output of **Evaluation workflow** includes following **Artifacts** :
• Release Specifications
• Release Descriptions
• User manuals

7.10 Deployment

Objective : To lead towards the closure of the project

Table 7.10.1 : Activities and Outputs of the life cycle phases in 'Deployment Workflow'

Life Cycle Phases	Activities	Outputs
Inception	• Analyze user responses • Map the achieved goals stated in RDD and SRS	• Conformance of system completion •
Elaboration	• Define user manual • Make appropriate changes based on final completed product	• Completed User manuals
Construction	• Release manual to customers • Release implementation documents for reference	• Handing over the system and getting user acceptance
Transition	• Handover the software product, user manuals, and maintenance plan to the customer. • Handover test plans, test results, action taken details. • Conduct closing exercise.	• System closure as per the contract.

Ultimate output of **Deployment workflow** includes following **Artifacts :**
• Deployment Set

7.11 Iteration Workflows

Iteration workflow describes the sequential set of activities in various proportions and also states where the iteration is located in the project development life cycle. Iteration workflow is based on 'iteration planning' determined by *critical use cases* and *high risk items* and *decreasing in priority* as we progress across the development life cycle.

Example : RUP (rational unified process) is *use-case* driven. The various operational workflows that we have studied in the previous sections of this chapter are - management, requirements, design, implementation, assessment (evaluation), deployment and development of infrastructure. Management is actually a planning and control workflow that manages both project and product life cycle. When these workflows are executed in a systematic manner, they result in an evolutionary software product development. This is achieved by the iterative process of ideas - design - tests - component delivery under the seven project management workflows as shown in the Fig. 7.11.1;

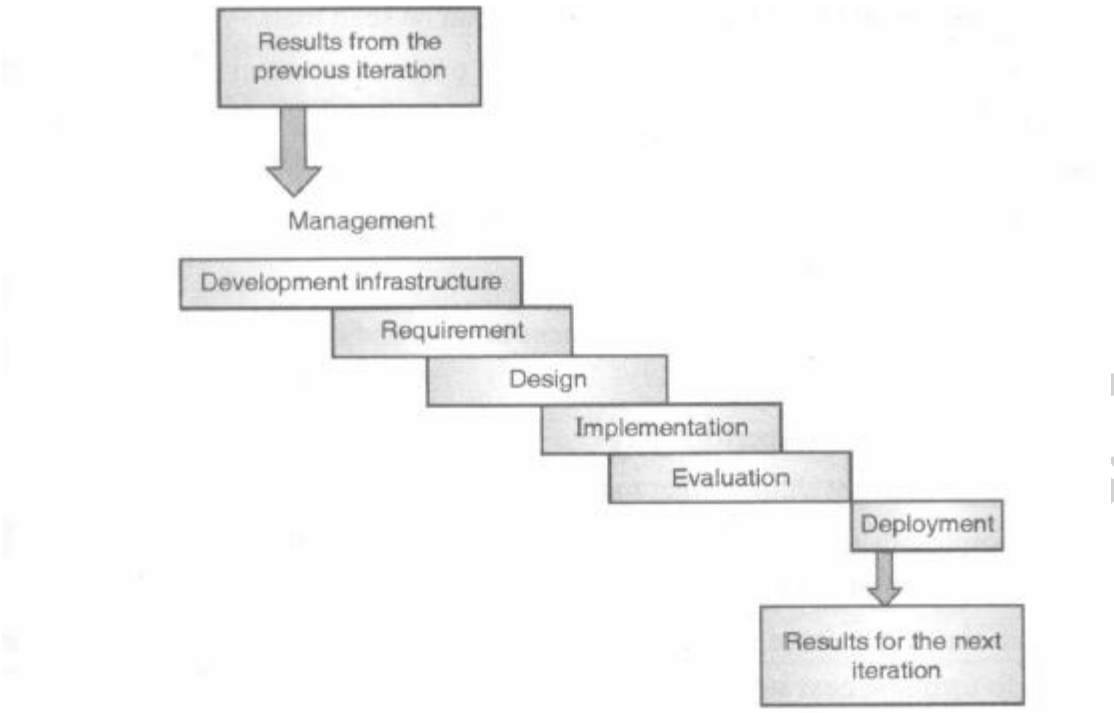


Fig. 7.11.1 : Iteration Overlaps over Project Development Life Cycle (scan)

The below figures represent *individual* iteration workflows;

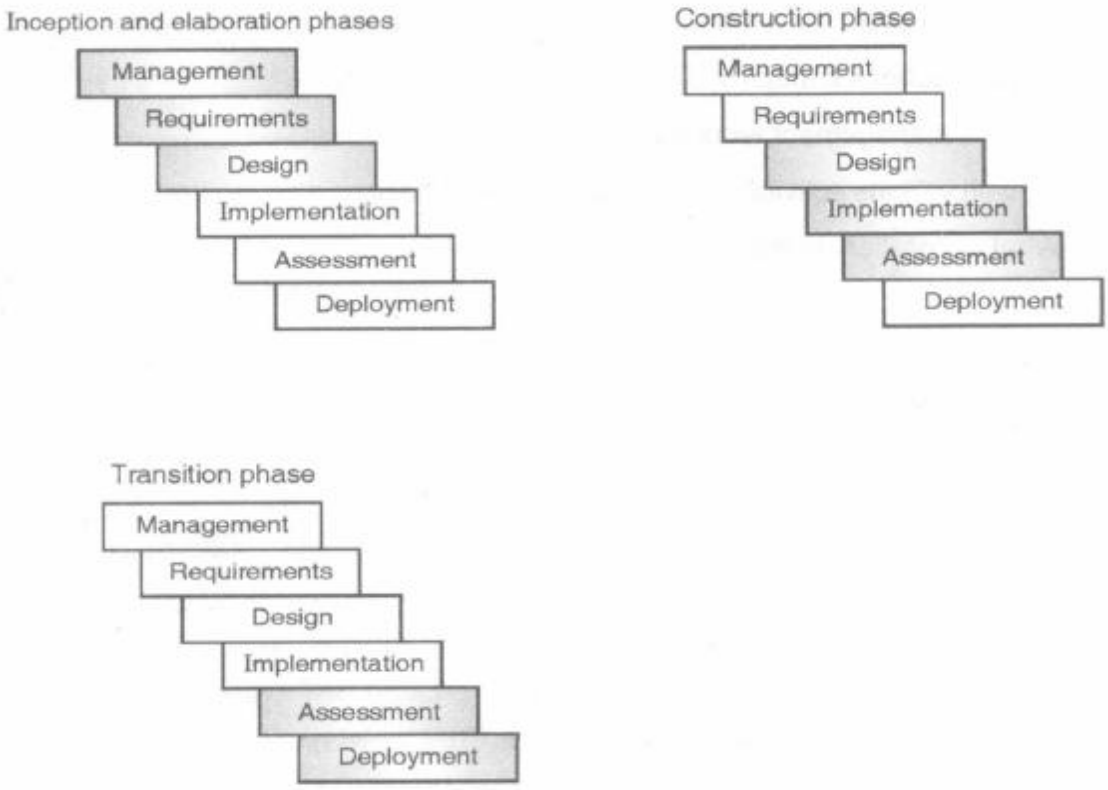


Fig. 7.11.2 : individual iteration workflows

From the above individual workflow figures, it can be seen that;
• Iterations in Inception and Elaboration focus on project management, requirements, and design activities.
• Iterations in Construction focus on design, implementation, and assessment.
• Iterations in Transition focus on assessment and deployment.

Characteristics of Iteration Workflow's :
• The iterations include continuous assessment of risks and results that describe :
o requirements
o design features and performance
o reviews and any changes that may occur
• Iterations represent the state of overall system architecture and also the progress status of complete deliverable system.
• Increment states the current work in progress that will be merged with the previous iteration to form the next iteration.

Review Questions
Q. 1 What is difference between old way and new way of process workflow ?
Q. 2 Explain Boehm's 7 major workflows.
Q. 3 What is management workflow ?
Q. 4 Explain various activities at various phases of design.
Q. 5 What is Iteration workflows ?

8. CHECKPOINTS

Syllabus :
Checkpoints of the process : Major mile stones, Minor Milestones, Periodic status assessments.

8.1 Evaluation of Workflow Process

A unified approach of software project management framework suggests 3 checkpoints so as to evaluate and assess the workflow process outputs.

(1) Major milestones
(2) Minor milestones
(3) Status assessments

Milestone :
• It is a well-defined 'measure of an activity'. It measures (evaluates) an activity and its result with maximum certainty.
• Milestones can be set in stakeholder meetings to discuss plans and progress, in defining the scope, in achieving the test results and etc.
• By setting milestones in the stakeholder meetings, the project manager :
o Coordinates stakeholders' (developers, customers and users) expectations and achieves a win-win agreement on the requirements, the design, and the plan,
o Coordinates the related artifacts to a reliable, consistent and balanced state
o Identifies the risks, issues, and also the out-of-tolerance conditions
o Conducts a global assessment for the whole life-cycle process

8.1.1 Three Checkpoints

Now, we will see in brief about the three checkpoints i.e. the '*three types of joint management reviews'* conducted throughout the process:

1. Major milestones :
• They are measured at the end of each phase as planned.
• They are termed as system-wide reviews since they are conducted at the end of each of all the four phases (Inception, Elaboration, Construction and Transition) of the lifecycle process.
• It gives idea of overall system issues, also synchronizes and coordinates the project management and the engineering perspectives and also verifies whether or not the aims of the phase are achieved.
• Major milestones use formal stakeholder-approved evaluation criteria and release Descriptions

2. Minor milestones :
• They are measured at the end of each processes and sub-processes within each phase.
• Minor milestones are set so as to confirm the completion of a tangible result.
• It includes iteration-focused events that are performed to conduct a detail review on the content of iteration and to authorize the continuation.
• Minor milestones use informal development-team-controlled versions of evaluation criteria and release description artifacts.

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

• It includes :
  - o Release specification that describes release plan and
  - o Evaluation criteria that results in the evaluation of the release.

3. Periodic Status assessments :

• They are performed to measure the process and its events and also it measures the completion of the goals of a process.

• Assessments are conducted periodically so as to provide the management with frequent and regular insights of the progress being made in the project development.

The below figure illustrates the typical sequence of checkpoints for a relatively large project.

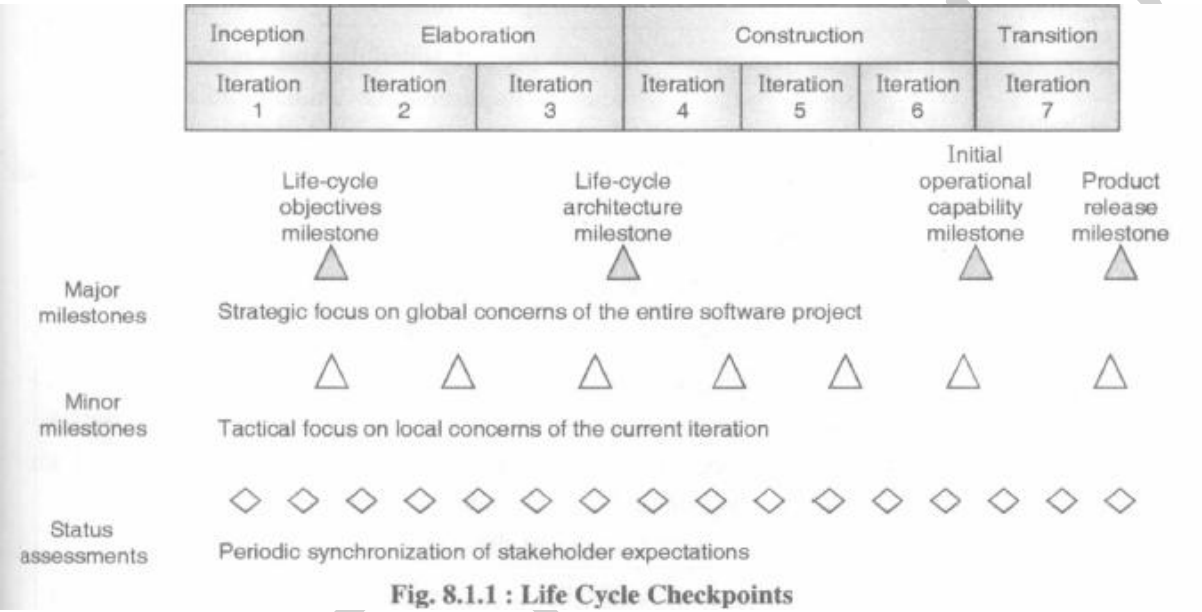


Fig. 8.1.1 : Life Cycle Checkpoints

8.2 Major Milestones

• In the figure 8.1.1, you will see four major milestones (lifecycle objective, lifecycle architecture, initial operational capability and produce release major milestones) occur at points between life-cycle phases.

• Major milestones are used in many different evolutionary process models such as the conventional waterfall model and so on.

• In an iterative model, it happens that different stakeholders have different expectations and therefore, to achieve the conformance of all these different stakeholders, some major milestones are set at the end of each of the four phases so as to achieve harmonious agreement among all stakeholders on the current status of the project. In the following section, we will see the different stakeholders who are involved in reviewing the major milestones of the project.

8.2.1 Major Milestones Reviewers

Here, we will see who the different stakeholders are and what their variant expectations are.

Table 8.2.1 : Major Milestone Reviewers

Stakeholders		Expectations
Customers	:	They are concerned with the project schedule and budget estimates, feasibility, risk assessment, requirements understanding, project progress.
Users	:	They are concerned with requirements consistency and their usage, potential to accommodate growth and quality features
Architects and systems engineers	:	They are concerned with the product line compatibility, any changes in requirements, trade-off analyses, accomplishment and consistency, risk assessments, quality, and usability features.
Developers	:	They are concerned with the sufficient details about the requirements and their respective usage scenario descriptions, frameworks for component selection and development, resolution of risks, product line compatibility, satisfactoriness of the development environment
Maintainers	:	They are concerned with the sufficiency of the product and related artifacts, project understandability, interoperability with existing systems, and satisfactoriness of maintenance environment
Others	:	There are also many other stakeholders who are concerned with what's happening in the project. These stakeholders may be regulatory agencies, independent verification and validation contractors, capital investors and sales and marketing teams

8.2.2 Status across Major Milestones

• There are in all four major milestones as following;

1. Life-cycle objective milestones
2. Life-cycle architecture milestones
3. Initial operational capability milestone
4. Product release milestone

.All these four major milestones occur at the end of each phase so as to achieve the conformance of the various stakeholders.

The below table depicts the status of development plans, status of customer requirements and status of product across the major milestones.

Table 8.2.2 : Status of Plans, Requirements and Products across the Major Milestones

Milestones	Plans	Understanding of problem space (Requirement)	Solution space progress (Software product)
Life-cycle objectives milestone	Definition of stakeholders responsibility Low-fidelity life cycle plan. High fidelity elaboration phase plan.	Baseline vision, including growth vectors, quality attributes, and priorities. Use case model	Demonstration of at least one feasible architecture. Make/buy/reuse trade-offs Initial design model
Life-cycle architecture milestones.	High-fedility construction phase plan (bill of materials, labor allocation) Low-fidelity transition phase plan.	Stable vision and use case model. Evaluation criteria for construction releases, intial operational capability. Draft user manual	Stable design set Make/buy/reuse decisions Critical component protypes
Initial operational capability milestones	High-fidelity transition phase plan.	Acceptance criteria for product release Releasable user manual	Stable implementation set Critical features and core capabilities Objective insight into product qualities
Product release milestones	Next-generation product plan.	Final user manual	Stable deployment set full features compliant quality.

1. Life-Cycle Objectives Milestone :

• This milestone is set at the end of the inception phase.
• It includes :
o Detailed Plan for the elaboration phase
o Cost and Schedule Estimates
o Expected Benefits from the system
o Project Vision Statement
o Critical issues about the requirements and operational concept of the system

• It contains
o A draft architecture document.
o A prototype architecture demonstration.

• Goals:

o To authorize all the stakeholders to advance to the elaboration phase including the plan, cost and schedule estimates, and expected benefits.
o To achieve a successfully completed 'life-cycle objectives milestone' so that the project manager can get an authorization (or say a green signal i.e. the permission) from all the stakeholders to proceed towards the elaboration phase.

2. Life-Cycle Architecture Milestone
• This milestone is set at the end of the elaboration phase.
• It includes :
o Detailed plan for the construction phase
o Critical issues regarding the requirements and operational concept of the system
o Baseline architecture that consists of both a human-readable representation and a configuration-controlled set of software components that are captured in the engineering artifacts.
o Vision statement
o Software development plan
o Evaluation criteria for the next milestone i.e. for the initial operational capability milestone.

• It contains :
o A Presentation and Overview of the current status of the software project.
o A configuration-controlled set of all engineering data such as the set of software components for writing down in the engineering artifacts.
o An executable representation of project capability.

• Goals:
o To demonstrate an executable architecture to all stakeholders
o To achieve a successfully completed 'life-cycle architecture milestone' so that the project manager can get an authorization from all the stakeholders to proceed towards the construction phase. The below figure lists the technical data i.e. reviewed by the lifecycle architecture milestone.

(I) Requirements:
(a) Use case model
(b) Vision document (text, use cases)
(c) Evaluation criteria for elaboration (text, scenarios)(lI) Architecture :
(a) Design view (object models)
(b) Process view (if necessary, run-time layout, executable code structure)
(c) Component view (subsystem layout, make/buy/reuse component identification)
(d) Deployment view (target run-time layout, target executable code structure)
(e) Use case view (test case structure, test result expectation)
1) Draft user manual.
(Ill) Source and executable libraries :
(a) Product components
(b) Test components
(c) Environment and tool components

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

Preparations for Life-Cycle Architecture Milestone :
Definitions of critical use cases and prepared scenarios for evaluating architecture
• Demonstrations of baselined stable architectures
• Understandable profiles of outstanding risks
• Elaborated mitigation plans
• Defined development plans for remaining phases

Presentation agenda :
(I) Scope and objectives
(a) Demonstration overview
(II) Requirements assessment
(a) Project vision and use cases
(b) Primary scenarios and evaluation criteria
(III) Architecture assessment
(a) Progress
(1) Baseline architecture metrics (progress to date and baseline for measuring future architectural stability, scrap and rework)
(2) Development metrics baseline estimate (for assessing future progress)
(3) Test metrics baseline estimation (for assessing future program of the test team)

(b) Quality
(1) Architectural features (demonstration capability summary vs. evaluation criteria)
(2) Performance (demonstration capability summary' vs. evaluation criteria)
(3) Exposed architectural risks and resolution plans.
(4) Affordability and make/buy/reuse trade-offs.
(IV) Construction phase plan assessment
(a) Iteration content and use case allocation.
(b) Next iteration (s) detailed plan and evaluation criteria.
(c) Elaboration phase cost/schedule performance.
(d) Construction phase resource plan and basis of estimate.
(e) Risk assessment.
Demonstration Agenda :
(1) Evaluation criteria
(2) Architecture subset summary
(3) Demonstration environment summary
(4) Scripted demonstration scenarios
(5) Evaluation criteria results and follow-up items

3. Initial Operational Capability Milestone :
• This milestone is set at the end of the construction phase.
• Goals:
o To assess the readiness of the software so as to proceed towards product transition onto customer/user environments.
o To authorize the beginning of acceptance testing. Acceptance testing is performed iteratively in iterations.

• It includes :

o Installation instructions and critical issues
o software version descriptions
o Operator manuals
o Support for test environment, and test software.

4. Product Release Milestone
• This milestone is set at the end of the transition phase.
• Goals:

o To assess the accomplishment of the software
o To assess the software transition
o To review the results of acceptance testing
o To address all the open issues
o To review all the software quality metrics and assess whether the reviewed quality features sufficient for software transition onto the support organization.

• It contains :
o Acceptance test outputs
o Open issues
o Installation issues and
o Support issues.

8.3 Minor Milestones
In most of the iterations where one - six months duration is required, there only two types of minor milestones are required.

1. Iteration Readiness Review
• This is an informal milestone i.e. set at begin of each iteration.
• It conducts detail review of iteration plan and its evaluation criteria.
o Early iteration focuses on design and analysis, experimentations, and risk assessment.
o Later iteration focuses on completeness, consistency, usage, and change management.

2. Iteration Assessment Review :
• This is also as informal milestone i.e. set at end of each iteration.
• It assesses the degree to which the iteration has achieved its objectives and satisfied its evaluation criteria,
• It reviews the iteration and test results. And on the basis of these results, amount of required rework is determined, and the impact of iteration results on the plan for subsequent iterations is also reviewed.

Other Minor Milestones :
• Test readiness reviews
• Test results
• Special issues.

The below figure depicts the various minor milestones that need to be considered while planning a project.

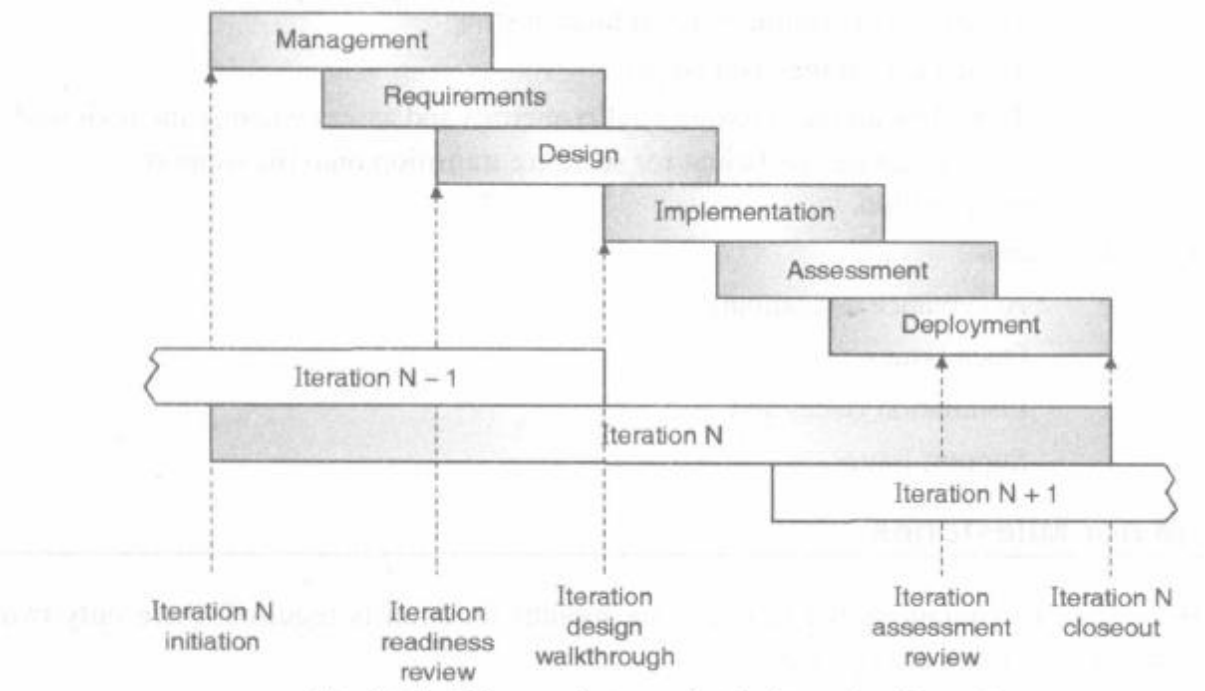


Fig. 8.3.1 : Minor milestones in the iteration life cycle

Periodic Status Assessments

• These checkpoints are the management reviews that are conducted at regular intervals like monthly or quarterly during the project development so as to address the project progress and to identify its quality.

• In short, such an assessment provides project snapshots.

• This is useful so as to ensure continuous attention over project dynamics and maintain open communications among the various stakeholders.

• Status assessments are necessary for forcing the management to focus on the quality and progress of the project.

• Status assessments are also used for making project-to-project comparisons, and distributing best practices within the organization.

Advantages :

• Status assessments provide a mechanism for open addressing and communicating to stakeholders, resolving management and technical issues, and also resolving project risks.

• Status assessments can derive the objective data directly from on-going activities and from the evolving product configurations. Status assessments provide a mechanism for propagating the project progress, its quality trends, practices, and experience information among all the stakeholders in an open address forum.

• Status assessments are necessary for forcing the management to focus on the quality and progress of the project and its dynamic priorities.

Table 8.4.1 : Template for Status Assessment Reviews

Topic	Content
Personnel	Staffing plant vs. Actual Attritions, additions
Financial trends	Expenditure plant vs. Actual for the previous, current, and next major milestones.
Top 10 risks	Issues and criticality resolution plans Quantification (cost, time, quality) of exposure
Technical progress	Configuration baseline schedules for major milestones Software management metrics and indicators Current change trends Test and quality assessments
Major milestones plans and results	Plan, schedule, and risks for the next major millstones pass/fail results for all acceptance criteria
Total product scope	Total size, growth, and acceptance criteria perturbations

Review Questions

Q. 1 What do you mean by milestone ?

Q. 2 Explain three checkpoints in short.

Q. 3 State and explain various stakeholders and their expectations for major mild stone reviewers.

Q. 4 What are the types of major milestone ?

Q. 5 Explain lifecycle objective of milestone.

Q. 6 What is a difference between iterative readiness review and iteration assessment review ?

Q. 7 What is periodic status assessment and what are it's advantages ?

9. ITERATIVE PROCESS PLANNING

Syllabus:
Work breakdown structures, planning guidelines, cost and schedule estimating,Iteration planning, process, pragmatic planning.

9.1 Work Breakdown Structure

A good Work Breakdown Structure (WBS) and its synchronized management with the process framework are necessary for the success of software project development. A good WBS is developed based upon few characteristics such as - way of conducting project management, culture followed in the organization, expectations of customers, and economical constraints.
A WBS decomposes the project plan into various unique work tasks. And various factors that lead to such decomposition are:

• product components
• functions
• organizational units
• life-cycle phases

A WBS generally consists of following information structure:
• explanations about all major works
• Clear decomposition of task responsibilities
• Framework for scheduling, budgeting, and tracking the expenditures

9.2 Conventional WBS Issues

1.Conventional WBS has three major drawbacks i.e. they undergo three basic flaws:
Conventional WBS is developed based upon the product design.
Below is an example of a conventional WBS that is developed based upon the product subsystems and then later on decomposed into minor components of each subsystem.

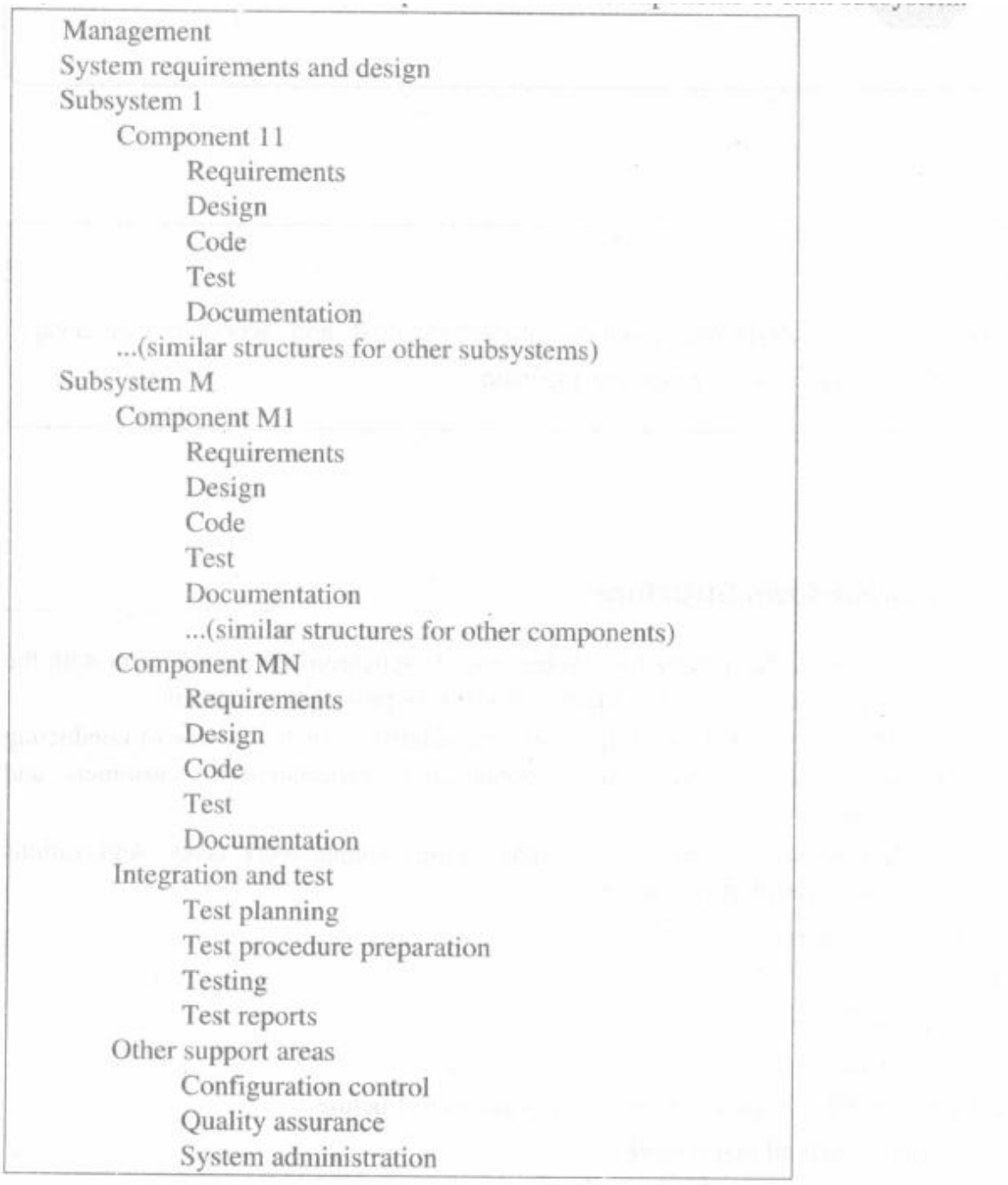


Fig. 9.2.1 : Conventional work breakdown structure, following the product hierarchy

1. Conventional WBS is planned so as to define the budget either in detail or in brief. It is generally observed that, large software projects are over planned and small software projects are under planned. But the basic problem with over planning i.e. planning in too detail is that these details do not grow accordingly with the level of reliability.
2. Conventional WBS is project-specific and fails to incorporate cross-project comparisons. There is no standard WBS structure define and therefore, it becomes difficult to include

comparisons of different plans, budget, schedules, various organizational efficiencies and their cost trends, productivity trends, and quality attributes inculcated across various projects.

9.3 Evolutionary Work Breakdown Structures

Evolutionary WBS must organize the planning elements around the *process framework* rather than the *product framework.* Thus WBS organizes the planning elements in the following levels of hierarchy :

Level 1 : Elements are defined in various *workflows* such as the management, environment, requirements, design, implementation, assessment (evaluation), and deployment workflows.

Level 2 : Elements are defined in all the four *phases* of the life cycle i.e. the inception, elaboration, construction, and transition phases

Level 3 : Elements are defined in all the *activities* that derive the artifacts of each phase. A WBS that is consistent with the workflows, phases and activities of the process framework is shown in the figure below:

This structure describes how the elements of a process framework are integrated into a project plan.

The WBS incorporates a framework in order to estimate the costs and schedule of each element by allocating them across the project development company and by racking its expenditures.

A.	Management
AA	Inception phase management
AAA	Business case development
AAB	Elaboration phase release specifications
AAC	Elaboration phase WBS baselining
AAD	Software development plan
AAE	Inception phase project control and status assessments
AB	Elaboration phase management
ABA	Construction phase release specifications
ABB	Construction phase WBS baselining
ABC	Elaboration phase project control and status assessments

- AC Construction phase management
  - ACA Deployment phase planning
  - ACB Deployment phase WBS baselining
  - ACC Construction phase project control and status assessments
- AD Transition phase management
  - ADA Next generation planning
  - ADB Transition phase project and status assessments

B. Environment

- BA Inception phase environment specification
- BB Elaboration phase environment baselining
  - BBA Development environment installation and administration
  - BBB Development environment integration and custom toolsmithing
  - BBC SCO database formulation
- BC Construction phase environment maintenance
  - BCA Development environment installation and administration
  - BCB SCO database maintenance
- BD Transition phase environment maintenance
  - BDA Development environment maintenance and administration
  - BDB SCO database maintenance
  - BDC Maintenance environment packaging and transition

C. Requirements

- CA Inception phase requirements development
  - CAA Vision specification
  - CAB Use case modeling
- CB Elaboration phase requirements baselining
  - CBA Vision baselining
  - CBB Use case model baselining
- CC Construction phase requirements maintenance
- CD Transition phase requirements maintenance

D Design

- DA Inception phase architecture prototyping
- DB Elaboration phase architecutre baselining
  - DBA Architecture design modeling
  - DBB Design demonstration planning and conduct

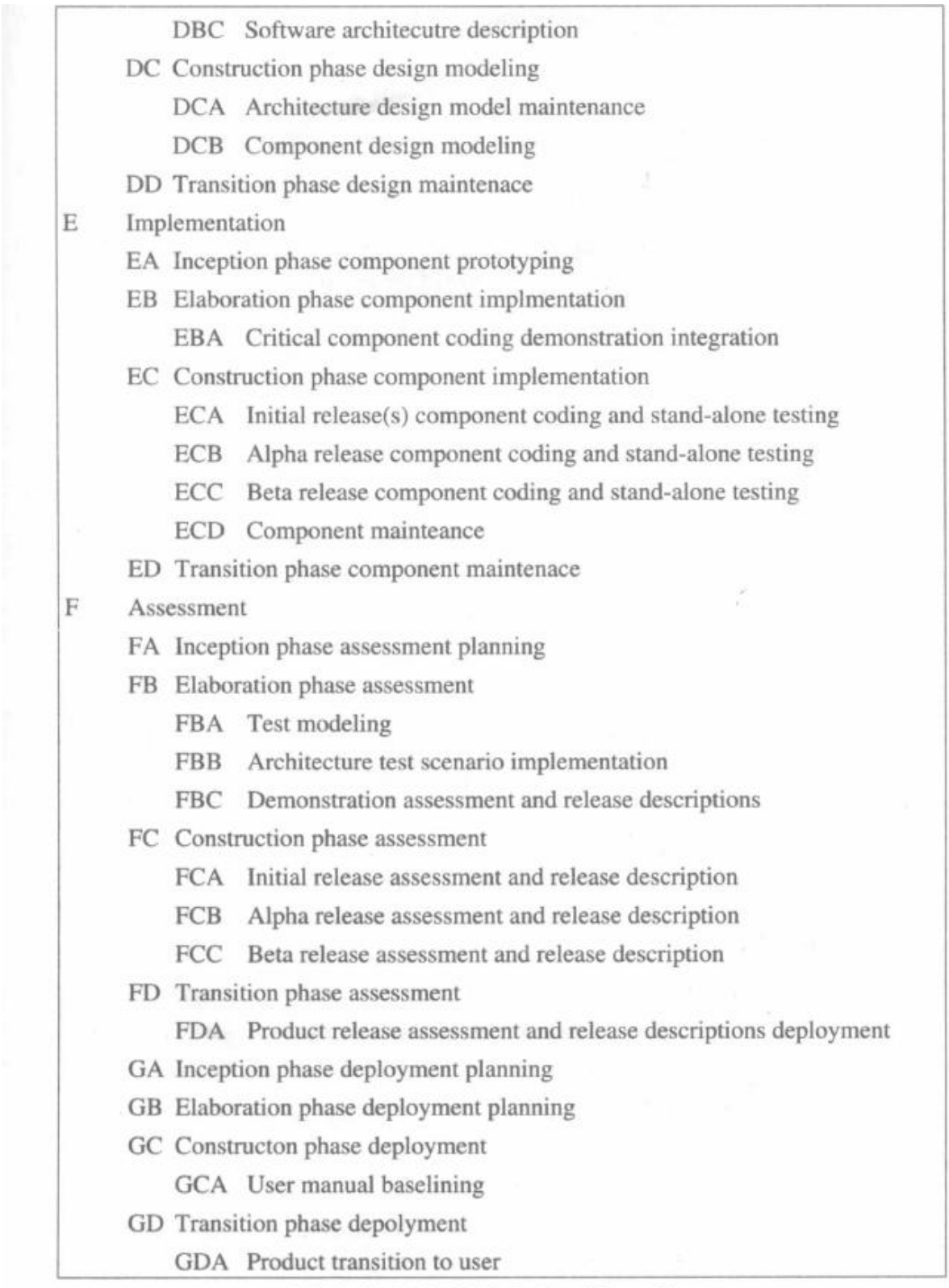


Fig. 9.3.1: Default Work Breakdown Structure

needs of that particular project.

• **Scale** : Bigger projects have more levels of **hierarchy** and substructures.
• **Organization's Structure** : Software projects that comprise of subcontractors or multiple organizational entities introduce few constraints that are helpful in various WBS allocations.
• **Level of custom development** : Focus varies upon the requirements, design, and implementation workflows as per the features of the project.
• **Business context** : There may be few commercial software products that are delivered to a broader category of customers. Such projects require much more detail description of the substructures for their deployment element.
• **Previous experiences** : Very rarely, it happens that a project needs to be started from scratch i.e. from a clean slate. Else, most of the projects are developed as new versions of legacy systems with an improved WBS.
A good WBS provides an insight into important features, function priorities and overall structure of the project plan.
Also, a WBS provides a reliability of planning elements ranging right from rough planning packages (such as rough budget estimation) to fully planned activities (well-defined budget and continuous assessment of actual v/s planned budget).

Inception		Elaboration		Construction		Transition	
WBS element	Fidelity	WBS element	Fidelity	WBS element	Fidelity	WBS element	Fidelity
Management	High	Management	High	Management	High	Management	High
Environment	Moderate	Environment	High	Environment	High	Environment	High
Requirements	High	Requirements	High	Requirements	Low	Requirements	Low
Design	Moderate	Design	High	Design	Moderate	Design	Low
Implementation	Low	Implementation	Moderate	Implementation	High	Implementation	Moderate
Assessment	Low	Assessment	Moderate	Assessment	High	Assessment	High
Deployment	Low	Deployment	Low	Deployment	Moderate	Deployment	High

Fig.9.3.2 : Evolution of planning fidelity in the WBS over the life cycle

9.4 Planning Guidelines

Software developing organizations work on a wider range of application domains. And, it is risky to make specific project plans independent of project context. Therefore, to avoid these risks, two simple planning guidelines should be adopted for developing a good project plan.
1. Default Web Budgeting: Roughly, allocate costs for the first-level WBS elements.

Table 9.4.1 : Web budgeting

First Level WBS Element	Default Budget
Project Management	10%
Project Environment	10%
Project Requirement	10%
Project Design	15%
Project Implementation	25%
Project Assessment	25%
Project Deployment	5%
Total	100%

2. Allocate the efforts and schedule across the lifecycle phases.

Table 9.4.2 : Distribution of efforts and schedules in each phase

Phases / Domain	Inception phase	Elaboration phase	Construction phase	Transition phase
Effort	5%	20%	65%	10%
Schedule	10%	30%	50%	10%

The values in both the above tables differ as per the requirements and constraints of the application.

Advantages of Planning Guidelines :

Using these two guidelines, it becomes easy to :

- Develop a staffing profile,
- Allocate staff resources
- Schedule the project
- Develop a WBS with task budgets and schedules relatively straightforward.

This type of top-down development planning is very useful and results in a baseline that can be supportive for further elaboration.

9.5 Cost and Schedule Estimating Process

Project plan is developed based upon two different perspectives :

I. Forward -looking:

- This perspective is also called as *top-down* approach.

• This approach begins with the requirements stage i.e. understanding the customer requirements - then based upon these requirements, a macro-level budget and schedule is derived - and at last, the planned elements are decomposed so as to define still more detailed budgets and accordingly sets the milestones.

• This perspective works in the following sequence:

1. Software project manager and his team define the overall size, process, environment, people, and quality of the proposed project.
2. A macro-level estimation of the total effort and schedule required for the proposed project is developed using a cost estimation model.
3. Software project manager decomposes the efforts into top-level WBS using the two planning guidelines described in previous section.
4. Developers working under project managers are given the responsibility of decomposing each of the WBS planned elements into much more lower levels based upon their top-level allocation and by considering the staffing method and assuming major milestone dates as the project constraints.

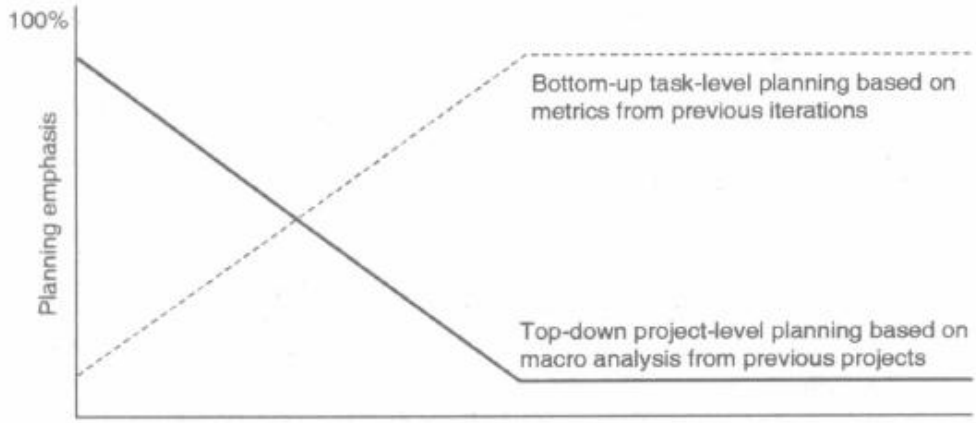
II. backward -looking:
• It is also known as a bottom-up approach.
• It begins from the end i.e. analyze the micro-level budgets and schedules - then sum up all these elements to form higher level budgets and intermediate milestones.
• This approach develops the WBS from lowest levels progressively to upward levels.
• This perspective works in the following sequence :
1. The lowest level elements are decomposed into detailed tasks. And the required budget and schedule for accomplishing these tasks is estimated by the responsible WBS manager.
2. Various such estimates are integrated collectively to achieve higher level budgets and milestones. The biases of individual estimators need to be homogenized so that there is a consistent basis of negotiation.
The top-down budgets and schedule milestones are compared with the budgets and schedule milestones of the bottom-up approach. The differences are assessed and refinements are made to achieve a balanced agreement between the top-down and the bottom-up estimates.

Forward looking v/s Backward looking approach (Top-down v/s Bottom-up approach):
• Estimating the budget and schedule using *top-down* approach increases the project management biases and yields an overly optimistic plan. Top-down approach is used mostly as a global assessment technique.
• But, estimating the budget and schedule using a *bottom-up* approach increases the performer biases and yields an overly pessimistic plan.
And these two perspectives or say, approaches should be used together, in an equal balance, throughout the life cycle phases of the project.
• In the engineering stage (inception and elaboration phases), the top-down perspective is recommended because in this stage, the depth of understanding is not to the mark and nor there is any stability in the task sequences which may else recommend bottom-up planning.
• In the production stage (construction and transition), only if one has enough previous experience and planning fidelity, only then the bottom-up perspective is recommended.

9.6 Iteration Planning Process

Planning defines the actual sequence of intermediate results. An evolutionary plan is very much necessary because it allows adjustments in build content and schedule along with the early assumptions that evolve into well-understood project circumstances.

Iteration allows complete synchronization across the project along with the global assessment of entire project baseline.



Engineering Stage		Production Stage	
Inception	Elaboration	Construction	Transition
Feasibility iterations	Architecture iterations	Usable iterations	Product releases

Fig. 9.6.1 : An Iteration Plan throughout the life cycle

• Inception iterations :

o It is an early prototyping activity that integrates the foundation components and results in an executable framework,

o The framework helps in detailing the complex use cases of the system,

o It describes the existing components, custom models and related requirements that in-tum describe the candidate architecture and develops a realistic business case and a suitable software development plan.

• Elaboration iterations:

o It results in an architecture that includes complete framework that is required for execution.

o And after the completion of architecture iteration, few complex use cases are demonstrated which include the following tasks:

1. initializing the architecture
2. injecting a scenario to force the worst-case data processing flow through the system
3. Injecting a scenario to force the worst-case control flow through the system.

• Construction iterations :

o Many of the projects include near about two construction iterations as follows :

1. alpha release
2. beta release.

• Transition iterations:

o Most of the projects involve single transition iteration i.e. a beta release into the final product.

A typical project may have following six iterations:

• One inception iteration - 1) architecture prototype

• Two elaboration iterations - 2) architecture prototype and 3) architecture baseline

• Two construction iterations - 4) alpha and 5) beta releases

• One transition iteration - 6) product release

A project that involves many stakeholders uses about nine iterations - 7) one additional inception iteration and 8 & 9) two additional construction iterations to achieve a total of nine iterations.

9.7 Pragmatic Planning

• Performing the iterations becomes easy if a plan is ready. While implementing an iteration 'n' of any phase, the software project manager simultaneously monitors and controls against the plan that was initiated in iteration 'n - 1 ' and the planning iteration 'n + 1'.

• A good project- management is an art of making trade-offs in the current and the next iteration plans based on the results of current and previous iterations.

• Success of every project is achieved only by good planning. That means, apart from poor architectures and misunderstood requirements, inadequate planning is also one of the most common reasons for project failures.

• A project plan defines how the requirements will be later transformed into a product considering the business constraints* And this product will be realistic, understandable by the stakeholders, and easily usable.

• Plans are not only useful for the project managers - As more open is the planning process and its results, more ownership will be attained among the team members. That means, bad and closed plans cause destruction and good and open plans encourages the teamwork.

Review Questions

Q. 1 What is the purpose of WBS?

Q. 2 List down and describe the conventional WBS issues.

Q. 3 Describe the levels of hierarchy in which the WBS organizes the planning elements.

Q. 4 List out the planning guidelines that need to be adopted for developing a good project plan.

Q. 5 What are the two approaches which allow estimating the required cost and schedule?

Q. 6 How do the top-down and bottom-up approaches differ?

Q. 7 Explain the iteration planning process.

Q. 8 What is the need of pragmatic planning?

10. Project Organizations and Responsibilities

Syllabus :

Line of Business Organizations, Project Organizations, Evolution of Organizations

10.1 Introduction

Organisation is an important part in the software Line - of - business as it fulfils the basic needs necessary to support the software development. Similarly, *Project Organization* is the large extend about the people involved in software development - Various different types of people come together and form a team for the project organisation and these teams are responsible for the work allocated to them. In general, project organisation binds some responsibilities to the team and allocatesome useful or decorative (designing and all) task towards that team members. This ensures the large architecture and small - small components also to complete whole project.

10.2 Project Management

Software Project Management (SPM) is an effort taken to develop a unique software product or service. It involves :

- definite timeframe
- budget
- definite specifications
- working in organizational boundaries

Reasons for Project Failure :

- Unclearly defined software requirements
- Inability to track and report the project's progress status
- Unmanageable risks
- Communication gap among the customers, developers, and users
- Use of unsuitable and poor technology
- Incapable to handle project's complexity
- Impractical expectations
- Inability to move beyond individual and personality conflicts
- Stakeholders' Politics
- Lack of project management

Software project management is :

• Creating an environment favourable to getting critical projects done.

• Regarding the activities that ensure that software is delivered on time and in accordance

with the requirements of the developing organisations as well as procuring organization.

• Regarding the activities involved in budgeting and scheduling the constraints that are

set by the developing organisation.

• After seeing the below figure, you will understand the activities carried out in the software project Management.

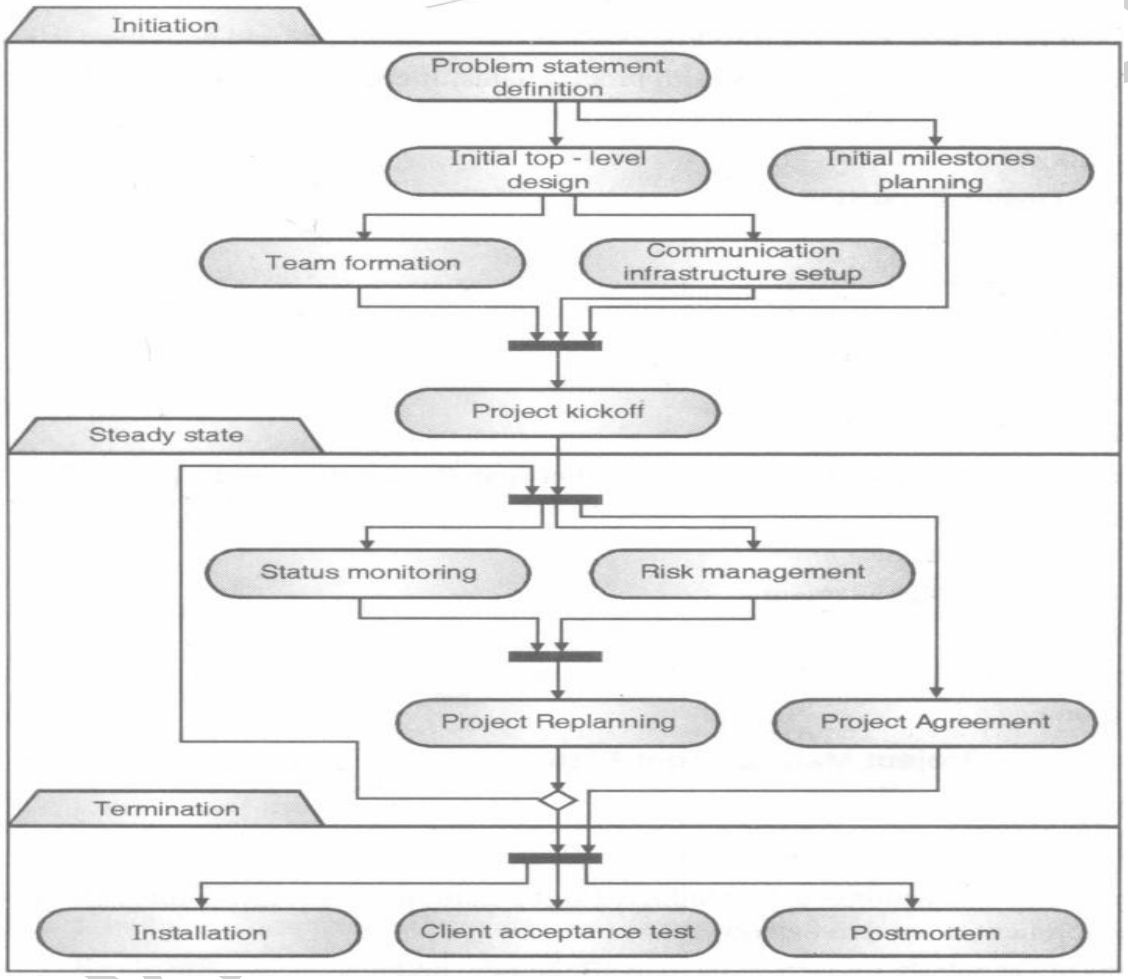


Fig. 10.2.1 : Software Project Management Activities Software Project Management is categorized into :

1. **Tasks** - It is the smallest unit of project management work i.e. again decomposed into sizes that allow monitoring. It involves planning and tracking.

Examples :

- Testing the components
- Writing the user manual
- Writing the MOM (minutes of meeting)
- Scheduling the code review
- Grouping the related tasks into hierarchical sets of functions and activities.

2. ***Activities*** - Major unit of work with precise dates. It consists of smaller tasks.

Examples :

- Planning
- Requirements Elicitation
- Requirements Analysis
- System Design
- Object Design
- Implementation
- System Testing
- Delivery

3. ***Functions* -** Activity or set of activities that span the duration of the project.

Examples :

- Project management
- Configuration Management
- Documentation
- Quality Control (Verification and validation)
- Training

10.2.1 Software Project Management Plan

Project Planning :

- Planning is the most time-consuming activity.
- This activity is continuously conducted right from the initial phase through the software development to system delivery.
- Plans are regularly revised so as to avail the new updates.
- Various types of plans such as for scheduling, budgeting and many such additional plans are developed so as to support the main software project plan.

A Software Project Management Plan :

- It is a major part of *project agreement.*

- It controls the software development process
- It describes the technical and managerial concepts involved in software development.
- It is a support for requirements analysis document.

Project Agreement is a document that defines :

- The scope (possible functions and limitations of the system), duration (time required to
- develop the system), cost (estimated budget) and deliverables for the project.
- System delivery dates, delivery location. Deliverables may be:
- Documents - user manuals
- Function descriptions
- Non-functional requirement descriptions
- Components' descriptions

Structure of a Software Project Management Plan :

- Front Matter
- Title Page
- Revision sheet (update history)
- Preface: Scope and purpose
- Tables of contents, figures, tables

1. Introduction:

- Project Overview
- It involves summarized description of project and product
- Project Deliverables »
- It involves the description of items to be delivered, their delivery dates and
- delivery location
- Evolution of the SPMP (Software Project Management plan)
- It involves descriptions of anticipated and unanticipated changes
- References
- It includes complete list of materials referenced in SPMP
- Keywords, Definitions and Acronyms

2. Project Organization :

1. Process Model

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

- Describes inter-relationships among various project elements regarding functions, activities, tasks, Milestones, Baselines, Reviews, Work breakdown structure, Project deliverables and Sign-offs.

2. Organizational Structure

- Describes internal management and organizational chart

3. Organizational Interfaces

- Describes relations with other entities

4. Responsibilities

- Describes the major functions and activities
- States responsibility of each team member as who is in-charge of which activity

3. Managerial Process :

Management Objectives and Priorities - Philosophy, goals and priorities

2.Assumptions, Dependencies, Constraints. Assumption may be that the security will not be addressed

- Dependency may be that the automatic code generation facility in the CASE tool depends on JDK.
- Constraints may be that the length of the project is 3 months, limited amount of time to build the system or the project consists of beginners and so, it will take time to learn how to use the tools.

3.Risk Management

- Identifying, assessing, tracking
- contingencies for risks

Example :

Risk : Members in key roles drop the course.
Contingency : Roles are assigned to somebody else. Functionality of the system is renegotiated with the client.
Example :

Risk : The project is falling behind schedule.
Contingency : Extra project meetings are scheduled.

4.Monitoring and Controlling

- It involves the reporting mechanism of information flows and code reviews

5.Staffing Plan

- It describes the required skills needed for project management.

6.Technical Process :

- Methods, Tools and Techniques - Computing system, development method, team structure, Standards, guidelines, policies.
- Software Documentation - Documentation plan, including milestones, reviews and baselines.
- Project Support Functions - Plans for functions (quality assurance, configuration management).

5. Work Elements, Schedule, Budget:

1. Work Packages (Work breakdown structure) - Project decomposed into tasks; definitions of tasks.

2. Dependencies - defines relations among the functions, activities and tasks in a priority.

3. Resource Requirements - lists the estimations of the resources such as development staff (personnel), development time, special hardware and support software, if any.

4. Budget and Resource Allocation - defines the costs related with functions, activities and tasks.

5. Schedule - states the deadline, dependencies and the project milestones.

Qualities o f a Good Project Planner:

- Writes a project plan based upon his experiences on previous projects.
- Tracks the activities and their duration
- Cross-checks the difference between planned and actual performance
- Does post-mortem to make him-self sure by asking for feedback from the developers,
- and by writing down about what can be improved and so on.

10.2.2 Project Scheduling

The *project schedule* is a calendar that is used to associate the tasks to be performs with the resources that will perform them. Before a project schedule is estimated, the project manager designs a work breakdown structure (WBS) which is an attempt to estimate the time needed to implement each task and the resources that are available for accomplishing each task. Project Scheduling is dependent on project managers' intuition and experience.

Steps of Project Scheduling :

- Divide the project into various tasks and estimate the duration and resources required to
- complete each task.
- Arrange the tasks so as to make optimal use of workforce.

- Reduce the task dependencies so as to avoid delays that might be caused by some task i.e. waiting for another to complete.

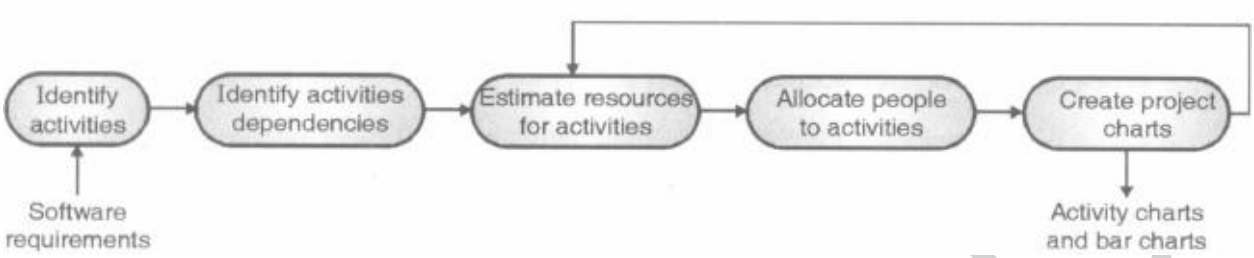


Fig. 10.2.2: Project Scheduling Process

Scheduling Problems :

- Detecting the complexity of the problems and accordingly estimating the cost of developing a solution is difficult.
- Achieved productivity is not proportional to the number of people working on it.
- Adding new staff in the late project development moves the delivery date further far because of the communication gap between old staff and new staff.
- Always unforeseen events happen in the project development and these need to be considered in the planning.

Scheduling Chart:

- Project scheduling involves preparing various graphical representations showing project activities, their durations and staffing.
- Graphical notations are used to illustrate the project schedule.
- Project scheduling is done by breaking down the whole development work into various tasks. Tasks must not be too small but should take about a week or two to accomplish them.

Activity	Duration (days)	Dependencies
T1	8	
T2	15	
T3	15	T1
T4	10	
T5	10	T2, T4
T6	5	T1, T2
T7	20	T1
T8	25	T4
T9	15	T3, T6
T10	15	T5, T7
T11	7	T9
T12	10	T11

- Fig: Project Scheduling - Project Breakdown into Tasks
- Following Activity chart describes the task dependencies and their critical path.

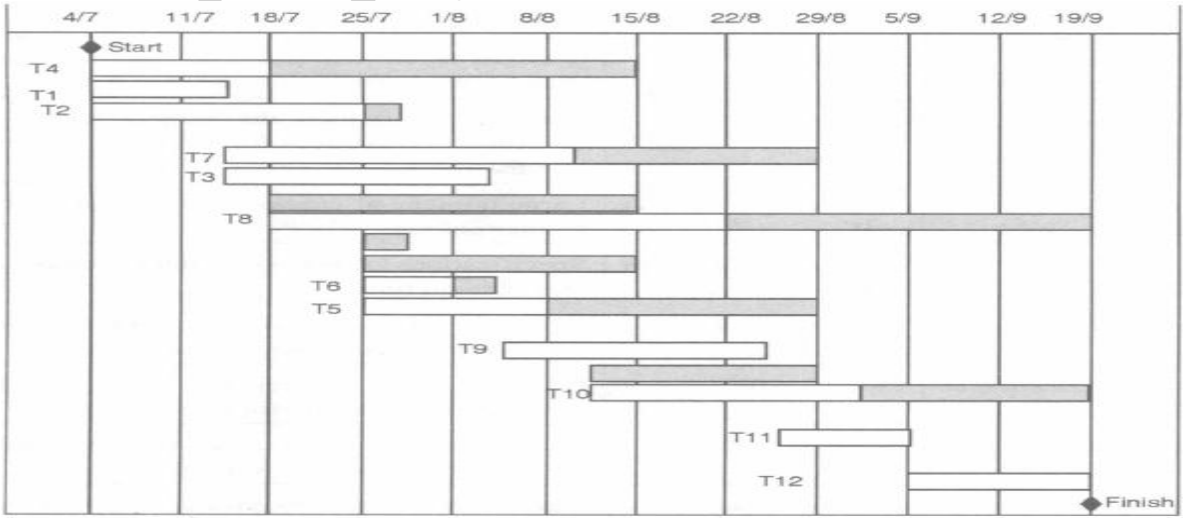


Fig. 10.2.3 : Project Scheduling - Activity Chart

10.2.3 Risk Management

- A risk is a probability of an occurrence of some adverse circumstance.
- Risks are categorized into three paths:
- Important Risks but not resolved
- Unimportant Risks that are not resolved and later ultimately change into Important Risks.
- Unidentified Risks that later become Important Risks

Types of Risks :

- *Project risks* affect project schedule and resources;
- *Product risks* affect product's quality and performance,
- *Business risks* affect development organisation and procuring organization.

Table 10.2.1: Software Risks

Reason of Risks	Risk Type	Description of Risks
Change in Staff	Project risk	Experienced and skilled staff leaves the project or organization before it is finished.
Change in Management	Project risk	Changes in various priorities lead to various changes in the Organisation.
Hardware unavailability	Project risk	Hardware on which the project has to be developed or deployed for test is not delivered on time.
Change in Requirements	Project and Product risk	Large number of changes in the requirements in later phases.
Specification delays	Project and Product risk	Specifications of necessary interfaces are not available on time.
Underestimating the size	Project and Product risk	The size of the system is underestimated which effects the scope, cost and schedule.
Poor performance of CASE tools	Product risk	CASE tools necessary for project development don't perform as expected.
Change in Technology	Business risk	The technology on which the system is built is outdated by some new technology.
Competition of Product	Business risk	Another competitive product is marketed before our product delivery and deployment is completed.

Risk Management Process :

- It is about identifying risks to ensure that these risks do not turn into major threats.
- It is about identifying risks and deriving plans to reduce their effect on the project.

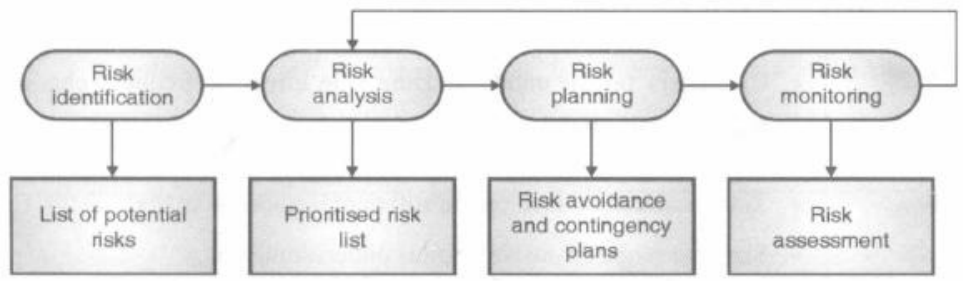


Fig. 10.2.4: Risk Management Process

- *Risk identification* - Identify project, product and business risks.

Table 10.2.2 : Risks and affects

Risk	Affects
Technology	• Database used in the system may be poor which cannot process as many transactions per second as expected. • Reusable software components might contain defects that limit the system's functionality.
Staff	• It is difficult to recruit skilled staff. • Key staff are unavailable at critical times may because they are ill or out of station. • Required training is not given to the staff.
Organizational	• One project is handled by number of managerial groups of the organization. • Organizational financial problems force cut-off in the project budget.
Tools	• Code generated by CASE tools is inefficient. • Not able to integrate the designs in CASE tools.
Requirements	• Changes in requirements lead to major changes in designs that lead to rework • Customers fail to understand the side effect of frequent changes in requirements on the project.
Estimation	• Time required to develop the software is underestimated. • Size and scope of the software is underestimated. • Defect Repair rate is underestimated.

- *Risk analysis* - Assess the likelihood and consequences of these risks,
  - Assess probability and seriousness of each risk,
  - Probability may be very low, low, moderate, high or very high,
  - Risk effects might be catastrophic, serious, tolerable or insignificant.

Table 10.2.3 : Risks, their probability and effects

Risk	Probability	Effects
Organizational financial problems force cut-off in the project budget.	Low	Catastrophic
It is difficult to recruit skilled staff.	High	Catastrophic
Key staff are unavailable at critical times	Moderate	Serious
Reusable software components might contain defects that limit the system's functionality.	Moderate	Serious
Changes in requirements lead to major changes in designs that lead to rework	Moderate	Serious
One project is handled by number of managerial groups of the organization	High	Serious

- *Risk planning* - Draw up plans to avoid or minimise the effects of the risk.
  - Consider each risk and develop a strategy to manage that risk,
  - Avoidance strategies-The probability that the risk will arise is reduced,
  - Minimization strategies - The impact of the risk on the project or product will be reduced.
  - Contingency plans - If the risk arises, contingency plans are plans to deal with that risk.

Table 10.2.4 : Risks and its strategy to overcome the risk

Risk	Strategy to overcome the risk
Organisational financial problems	Prepare a briefing document for senior management showing how the project makes an important contribution in achieving the goals of the business.
Requirement Change problems	Warn the customer regarding the potential difficulties caused by specification delays.
Staff Unavailability problems	Reorganise the development team so that there is more overlap of work and therefore, team members will more easily understand each other's jobs.
Defective Reusable components	Replace potentially defective components with components of known reliability.

- **Risk monitoring** - Monitor and control the risks throughout the project development,
  - Assess each and every identified risk regularly so as to check whether or not it is becoming less or higher,
  - Assess whether or not the effects of the risk have reduced,
  - Discuss each key risk in the management progress meetings. To come out from that risk, *line of Business Organization* is maintained.

10.3 Line of Business Organization

This is very important task of *allocation of responsibilities* which achieves process automation. This is an organizational task which is completed by either individual or several teams engaged in working on several tasks such as process definition and maintenance.

Fig. 10.3.1: Role of the Software line - of - business organization

10.3.1 Organization Manager

Organization Manager performs leading task in the organization. He performs various different roles to achieve success within the organization. He plays the roles of following leading activities;

- Leader
- Management administrative
- Professional Working tasks

Duties of Organization Manager

There are several duties of an Organization Manager which are listed below;

- Make plans for various processes.
- Build Organizational development strategy.
- Give directions, advice and counseling to the staff who are working for the process.
- Contribute in various tasks like Monitoring, coordinating, researching, analyzing, designing, coaching, providing recommendation, managing overall and etc.
- Play the role of 'Agent of Change' when required.
- Maintain various kinds of records.
- Support in data collection

10.3.2 Software Engineering Process Authority (SEPA)

The main task of Software Engineering Process Authority is to allow information exchange and provide project guidance for project practitioners. Organization general manager or any individual or a team representative performs the role of SEPA who keeps track of the process maturity and improvement in the future project process.

Duties of SPEA

There are several duties of a SPEA which are as follows;

- Initiate process
- Assess project or process in each mile stone.
- Responsible for process definition and maintenance - This duty involves modification, improvement and innovative tool insertion in the project.

10.3.3 Project Review Authority (PRA)

Project Review Authority is responsible for project compliance with all organizational and business units.

He is also responsible for meeting the requirements of a contract or some other project compliance standard.

Duties o f PRA

There are several duties of PRA which are as follows;

- Review customer commitment.
- Be devoted to organizational policies, organizational deliverables, financial performance, and other risks and accomplishments of the project task.
- Support other relevant teams.

10.3.4 Software Engineering Environment Authority (SEEA)

He performs the major role since whatever investment done on the single project SEEA can be targeted to achieve the return.

Duties o f SEEA

There are several duties of SEEA which are as follows;

- Automate the organization process - main task
- Maintain organization standard environment
- Train those people who are working on the project.

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

10.3.5 Infrastructure

It is a support system for the human, research and development (R&D) and other capital software engineering assets.

Components of the Infrastructure

1. Project Administrative work Infrastructure
- Time Management System
- Contract
- Pricing
- Rules and Regulations
- Legal information related to corporate world
2. Engineering skills centered Infrastructure
- Tools warehouse and maintenance department
- Proposal support system,
- independent R&D
3. Professional development:
- Organization of the departmental training camp
- Hiring employees
- Database maintenance skills
- Maintaining the library including literacy
- Publish technical support books

10.4 Project Organization

Several teams are involved in the project organizations which perform several different tasks of the project phase. One team consists of either individual or group of people. By default, project organization consists of ;

1. Project Management Team
2. Project Architecture Team
3. Software Project Development Team
4. Assessment Team

Each team takes care of the software *quality.* And, so as to achieve the quality, they check each and every milestone in the project. Only the *responsibility bearing* is different for different teams.

The following figure depicts the Software project organization in short;

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

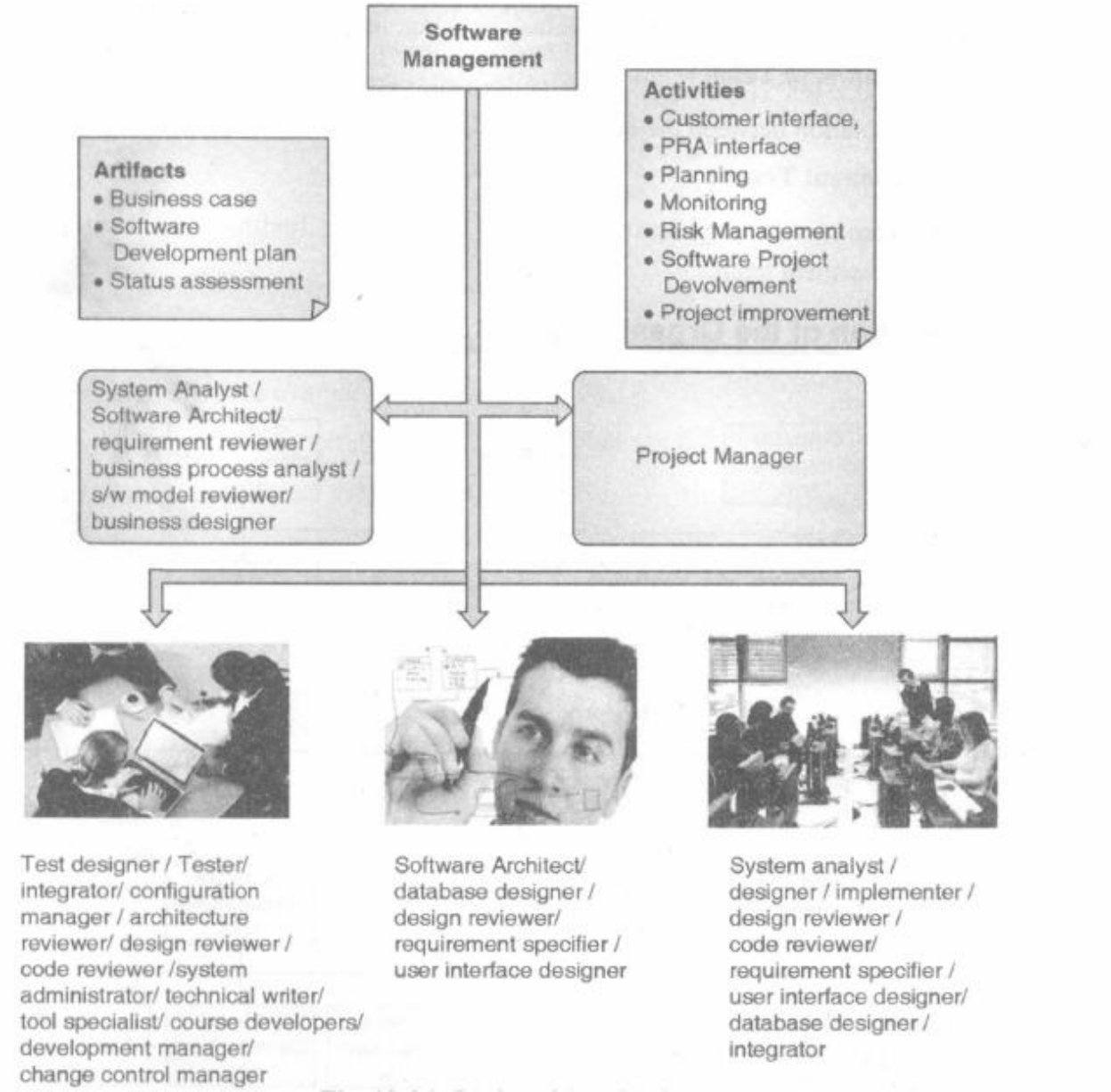


Fig. 10.4.1: Project Organization

Project Management Team :

- It is the top most management in the organization hierarchy.
- It is about overall management concerned with the project. They make an active participation in every section of the project development.
- *Production and management control* are the major responsibilities of the Project Management team.

Software Architecture Team :

It is responsible for staffing as well as it is engaged in the artful work such as the software designing, development, status checking, its assessment etc.

Software Development Team :

It makes relation between the components designing and its maintenance activity.

Software Assessment Team :

It takes care of software improvement. All the priorities finding done are cross-checked by the assessment team.

10.5 Evolution of the Organization

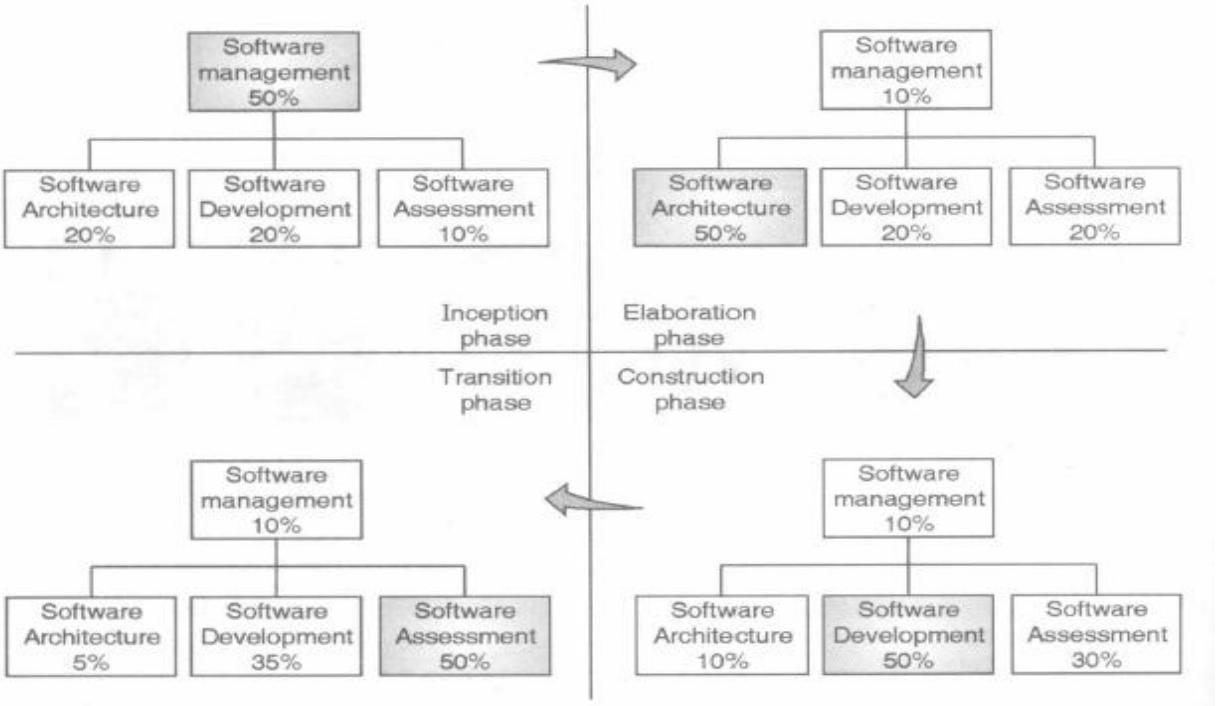


Fig. 10.5.1: Evolution of the Software Project Team over the Life Cycle phases

In the previous section, we have studied about different teams involved in the software project evolution. The evolution done in the each phase is as follows;

- **The Inception team :** The major task of the inception phase is planning and supporting

action to the other teams.

- **The Elaboration team** : The major task of elaboration phase is overall architectural designing. This phase takes help from the software development and software assessment teams to make architecture baseline for the project.
- **The Construction team :** The major activity of construction phase is software development. The partial assessment is also done in this project.
- **The Transition team** : This phase involves the customers in evolution who gives their feedback in the deployment activities.

Review Questions

Q. 1 Define software project management.

Q. 2 What are the reasons for project failures ?

Q. 3 What are the contains of project agreement ?

Q. 4 Explain in brief the overview of project planning.

Q. 5 What do you mean by project scheduling ? What are different steps involving in it ?

Q. 6 What is risk in project ? What are its types ?

Q. 7 How various risks handled in project ?

Q. 8 What is line-of-business ?

Q. 9 Write short notes on :

i) Organization manager

ii) SEPA

iii) PRA

iv) SEEA

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

11. Process Automation

Syllabus :

Process automation : Automation building blocks, The project environment.

11.1 Elements of Process Automation

- **Environment:** This is the prime element of Process Automation where the environment should be good and suitable for process growth. This is an artifact of the process.
- **Change Management Relation :** This relation is critical in nature in case of the iterative process and this is because change is essential thing in Process Automation. The changes are some times accepted and sometimes rejected also. So the proper change management is essential. And note that, costly changes are always rejected by the development department of the project organization.
- **Round Trip Engineering :** Since the project development department rejects the changes if it is expensive then in such cases, the collaboration of the Round Trip engineering and its environment promotes the freedom of change. This Change also gives technically skilled work.
- **Metric Automation :** As the changes are done, controlling these new changes is very important. Metric automation provides the essential environment for effective project control,
- **Role of External Stakeholders** : They try to interact with development team for the improvement at the time of process creation. They need to access the environmental resources to achieve such improvement. Process can be strongly created at this stage since some values are added in the process.
- **Process Level**: There are three levels of the process which are depicted in the following figure;

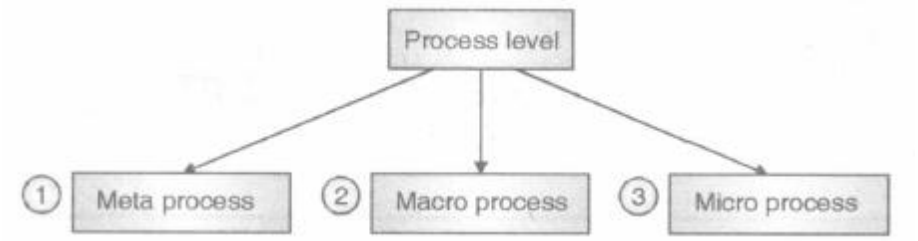


Fig. 11.1.1 : Process Types

- The above categorization of the process is done based on the degree of process automation ;

1. **Meta Process:** This automation in the process is done at the Line of business which we learn in the last chapter. The automation that supports at the Meta level is called 'infrastructure'.

2. **Macro Process:** This automation in the process is done at project development phase. The automation support at the macro level is called an environment.

3. **Micro Process:** This automation in the process is done at iteration phase. The automation support at the micro process level for generating artifacts is called as tool.

11.2 Automation Building Blocks

There are lots of tools available which are used to automate the software development process. Various software process development workflows use at least one tool.

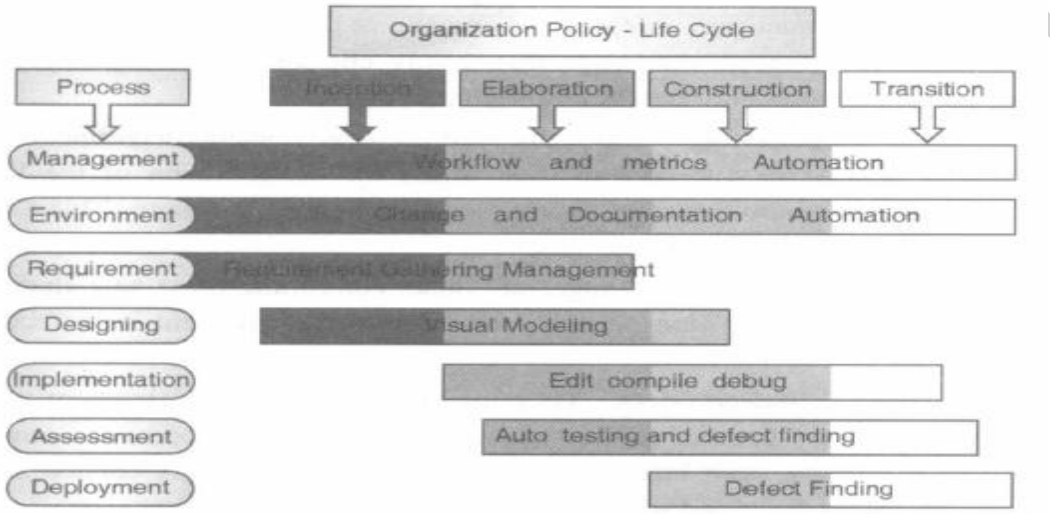


Fig. 11.2.1 : Automation and tool components involved in process workflow

Management:

Workflow is the process which contains small task links together. The automation in the tasks, resources and internal operations of the process is called Workflow Automation. Environment: Change management depends upon the new version of the product which includes configuration management. Requirements :

- The flow of the requirements finally reaches to smallest unit. System requirements decompose into subsystem, sub - system requirements are fulfilled by components while component requirements are generated by smallest unit.
- We can understand system Requirements through following figure.

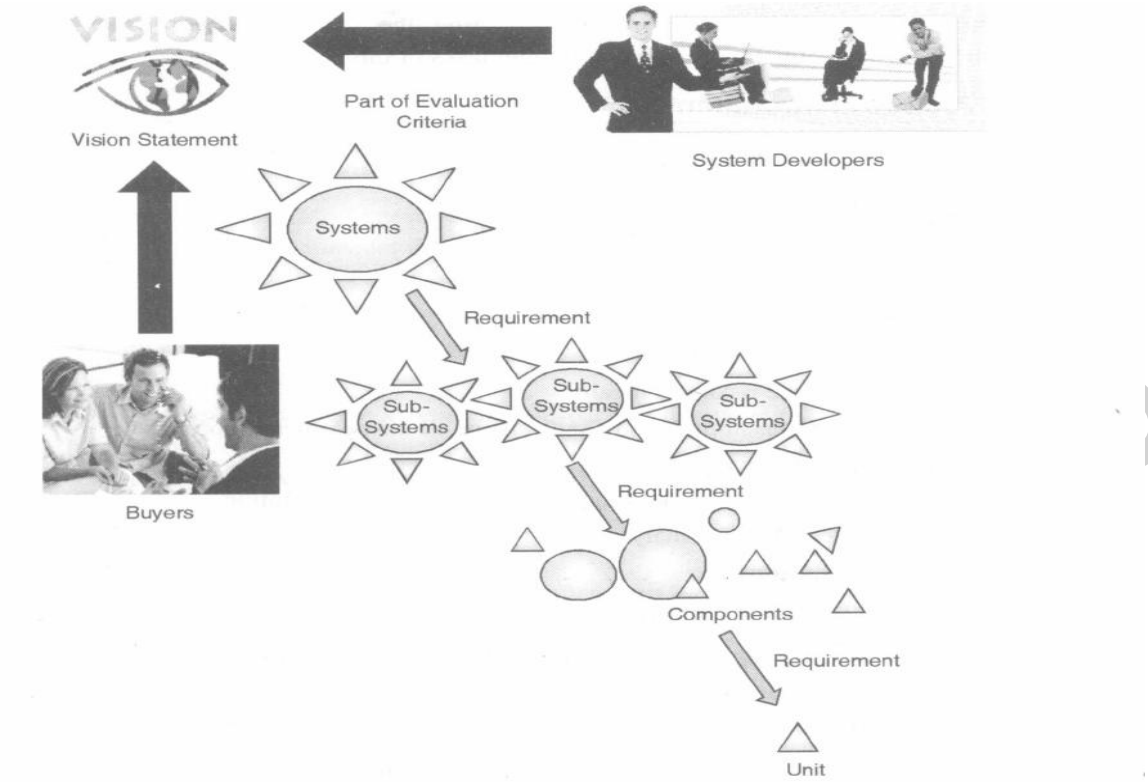


Fig. 11.2.2 : Requirement Flow

- Vision statement can be generated through the interaction amongst the System development group.
- Vision statement is accessible by the buyer of the System.

Design :

- The targeted tool for the design workflow is visual modeling.
- Generally this model is used to capture the design models which are further used to present it in human understandable format and finally translate it into source code.

Implementation :

This workflow is based on the productive iteration, which is based upon the programming environment include editing, compiling, debugging, linking, running etc. Assessment and Deployment: Assessment workflow depends upon testing the tool or system which is finally generated. Defect finding is one of the important tasks of this workflow.

11.3 Project Environment

Project Environment is categorized into three states :

1. Prototyping environment
2. Developing environment and
3. Maintenance environment

Prototyping Environment:

It includes following major activities :

- Analyses the risks technically
- Makes the feasibility study analysis for all the commercial products
- Reconfiguration
- Analyze the risks at the time of full system implementation
- Make requirement analyses and generate testing scenario for that etc.

Developing Environment:

It includes following major activities :

- Provide development tools for various process workflow
- Support Round trip engineering

Maintenance Environment:

- It is the subset of the development environment. It includes following major activities in-order to deliver the project's end product.
- In the environment aspect there are main four disciplines that are essential for management context & it further finalizes the success of a modern iterative development process.

1. Round Trip Engineering
2. Change Management
3. Infrastructure
4. Stakeholder Environment

11.3.1 Round T rip Engineering

• Today we get well-developed software product which is more reliable in each and every sector. When developers want to make consistency in the engineering artifact then Round - Trip Engineering comes in picture.

• Traceability is also maintained by the round trip engineering.

Definition:

The term used to describe the key requirement for system environment that support iterative development is known as Round Trip Engineering.

11.3.2 Change Management

- Change Management suggests fundamental changes in the iterative process such as planning. When we consider the iterative development then Change is the basic entity.
- In the Software Project Management, Change management is very important since customer always requires innovative things which will satisfy his requirements in all conditions. So, in

other words, we can say change management is iterative that means repetitive process in the planning.

- Change management is not done in one particular phase; rather it is done in several phases like requirement gathering, designing, implementation etc. That means all the activities which are performed in all the phases of software life cycle follow the change management process. Software Change Orders :
- Software Change Order (SCO) is the smallest unit in software department which works to create, modify or generate innovative components in the configuration baseline.
- Change primitives are described using the following major components;

1. Title
2. Description
3. Metrics
4. Resolution
5. Assessment
6. Disposition

- All these details are necessary for the modem software management. It is noted that one single SCO works with (or written) single component in the software.
- There are certain issues regarding the SCO such as;

Q. 1 What is major change about SC O written?

Q. 2 The new change is applied on exactly what?

(a) On subsystem

(b) On whole file

(c) A smallest unit or whole component

Q. 3 What is the type of change?

(a) Solution against the defect

(b) New innovative feature

(c) New creative change

- Now we discuss about the description of the SCO.

Title :

- Title is composed by the user of the software who is an external person and tells about his major problem regarding the software.
- Afterwards, this title is compiled by the originator and finalized by the Configuration Control Board (CCB).

Description :

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

- Description is composed of originators name, date is mentioned. When CCB finalized title, then addition to that CCB also assigned SCO identifier and version. These two fields are also mentioned in the description.
- The written description is in brief formats including all the details such as problem codes, view snapshots, error and alert message and few other relevant data by which change is stated in simple words.

Metrics :

- This is one of the important entities of the SCO.
- It is used for planning, scheduling and making quality improvement.
- There are few types of changes which are mentioned in the metrics field which are as follows;

Table 11.3.1 : Change type, its name and description

Change category	Name	Description
Type 0	Critical failure	Fixed before release Most of the critical cases it occurs
Type 1	Bug or defect	Very rarely this case happens These kinds of errors are problematic in critical use cases.
Type 2	Enhancement	If Type 1 / error is not resolved new enhancement done as change which is better for quality performance, testability, usefulness etc.
Type 3	Innovation	This change category is maintained for upgrade the software as per new requirements.
Type 4	Other changes	Documentation, version upgrading etc.

Example :

If exam mark sheet generation software has some changes then following change types occur.

Table 11.3.2 : Change type and example

Change Type	Mark sheet Generation Project
Type 0	Because of the virus or any other corruption data which is previously fed is lost.
Type 1	Entry of the marks in the database is maintained but the message as 'data successfully saved' or any other messages are not shown by the system
Type 2	College / university logo should be maintained as watermark background
Type 3	New subject addition entry must be provided in the software
Type 4	Previously MS – access supported only as the software backend but now the oracle 10g supports in Windows and Linux also

<Company Logo >

<Company Name and Address >

<Phone number and email id>

Phone number and email id>

Project Title:	Change Request Number:
Project Sponsor :	Change Requester:
Project Manager:	Date:

Description
<Change description mentioned here>

Metrics		
Category : ☐ Critical Failure ☐ Bug ☐ Enhancement ☐ Innovation ☐ Other Changes		
Breakage		
Rework		
Analysis		
Implement		
Testing		
Document		
Initial Estimate :	Actual Rework Expended :	

Resolution		
Analyst :		
Software Component		
Description		

Assessment		
Assessment Method	☐ Inception ☐ Analysis ☐ Demonstration ☐ Testing	
Testing person		
Platform		Date:
Other Description		

Disposition		
State		Priority
Acceptance Type		<Reason of acceptance>
Closure		

<Project Sponsor Name and Signature> <Project Manager Name and Date>

Fig. 11.3.1 : Components of SCO

Following are the various items which are mentioned in the metrics component;

Table 11.3.3 : Items with description

Item	Description
Breakage	In how much quantity change should be done is suggested by breakage.
Rework	Problems that arise in the change
Analysis	Time spent for understanding the changed problem
Implement	Necessary time for design and implement the solution against change
Testing	Test the created solutions
Document	Upgrade pending artifacts such as user handbook and releasing description

Resolution :

This field contains the analyst that means the person who is perform change, implement on that component which should be change, metrics and related description.

Assessment:

This filed contains the assessment methods such as inspection, analysis, demonstrate and testing etc.

Disposition :

- This field contains the state which is approved by the Configuration Control Board (CCB).
- There are following states in which change outlook is maintained;

Table 11.3.4 : Status and activities

States	Activities
Proposed	Change Resolution is written, but configuration control board (CCB) review pending
In progress	Assigned by the CCB and actively resolved by the them
Accepted	After view the change resolution, it is finalized and approved
Rejected	Change resolution is fully rejected with its justification; justification is given by the following main reasons; Whatever change resolution made, it is not that much level of problem Copied / Duplicate resolution The change resolution is made is out of date Change made by other SCO
Archived	Finalized and approved, but it is made to wait until new version releasing order is released
In assessment	Total change resolution is done, testing is in progress
Closed	Completely shut down along with the permission of the CCB.

Configuration Baseline :

- If newer version of the software product comes in the market then either it must be a totally newly launched product or must be an upgraded version of the previous product.
- If it is totally a new product then it is further tested and maintained properly in single unit.
- Configuration baseline is a collection of the software components and supporting documents which process all the work that are stated above.
- This baseline has two parts;

1. Internal testing release
2. External testing release

- If development organization makes testing for themselves then it known as internal testing. Testing done by configuration baseline before produce for the customer is known as a - (Alpha) testing. Project finalized and release from the configuration baseline to the user community is known (3 - (beta) testing.
- There are three levels are maintains by the configuration baseline which are **m ajor (M), minor (N) and interim (X)** which are number identifiers. All these three are releasing numbers. As the new product / component generates the major number identifier increments by 1 and so on while if the enhancement, quality feature is added in the existing product then minor identifier increments by the 1 and so on. An interim release identifier shows the transient development.

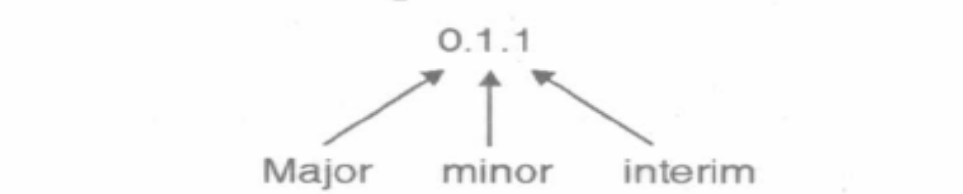


Fig. 11.3.2 : Baseline Release numbers

Following figure shows one of the examples of the releasing history of one of the projects;

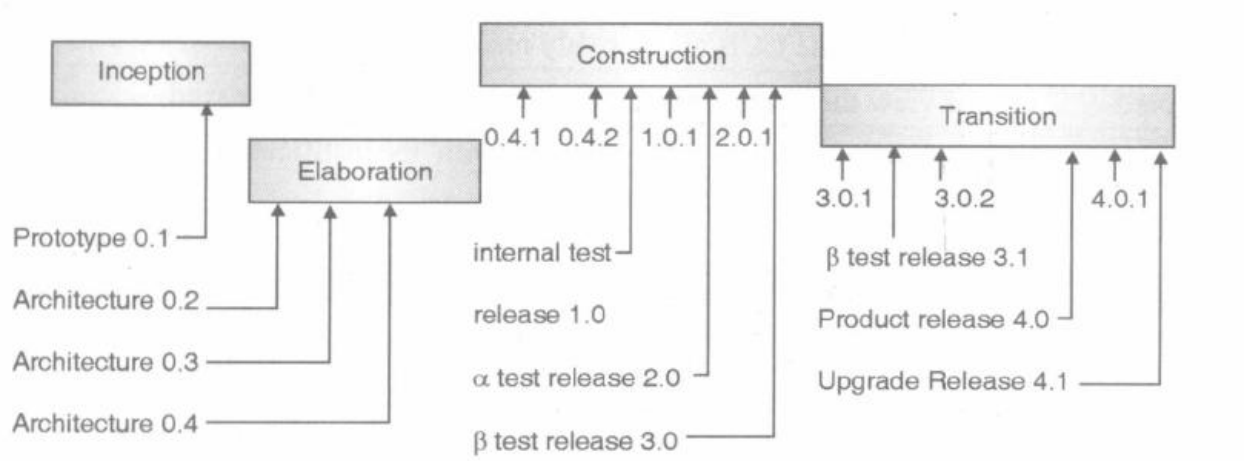


Fig. 11.3.3 : Historical Launch of software product with four different phase with baseline releasing numbers

Configuration Control Board :

- A decision authority that involves a group of people forms a team who give judgments, finalizations, and approvals on the configuration baseline contents. There are various software related people involved in the configuration control board (CCB), who are depicted in the following figure.
- As mentioned in the figure, CCB contains Software project manager, assessment manager, developer, team in-charge, customers, system engineers, user etc.
- Duties of the Configuration control board

1. Controlling software delivery system
2. Take various actions through CCB
3. Conduct meetings for various purposes
4. Online distribution of actions
5. Approval for software project release

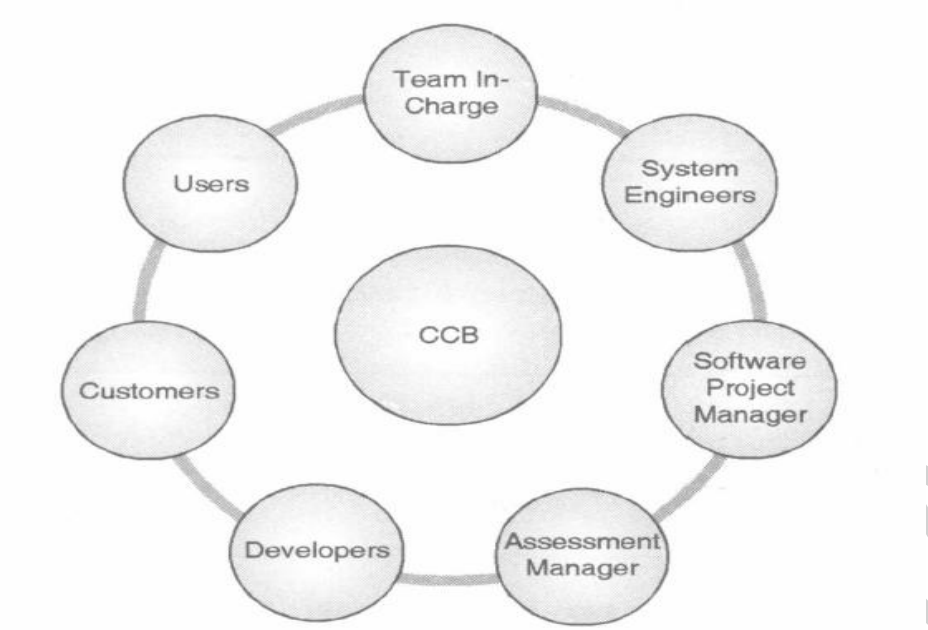


Fig. 11.3.4 : CCB Team

Review Questions

Q. 1 List out the elements of Process Automation

Q. 2 State and define the three levels of Process.

Q. 3 What are the building blocks of Process Automation. Explain with a neat diagram.

Q. 4 Describe round-trip engineering.

Q. 5 What is the need of change management?

Q. 6 Explain S C O and its various components/fields with a proper example.

Q. 7 What is the need of configuration baseline and describe CCB.

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

UNIT V

Project Control and Process instrumentation:

THE SEVEN CORE METRICS

Three are management indicators and four are quality indicators.

MANAGEMENT INDICATORS
- Work and progress (work performed over time)
- Budgeted cost and expenditures (cost incurred over time)
- Staffing and team dynamics (personnel changes over time)

QUALITY INDICATORS
- Change traffic and stability (change traffic over time)
- Breakage and modularity (average breakage per change over time)
- Rework and adaptability (average rework per change over time)
- Mean time between failures (MTBF) and maturity (defect rate over time)

Overview of the seven core metrics

METRIC	PURPOSE	PERSPECTIVES
Work and progress	Iteration planning, plan vs. actuals, management indicator	SLOC, function points, object points, scenarios, test cases, SCOs
Budgeted cost and expenditures	Financial insight, plan vs. actuals, management indicator	Cost per month, full-time staff per month, percentage of budget expended
Staffing and team dynamics	Resource plan vs. actuals, hiring rate, attrition rate	People per month added, people per month leaving
Change traffic and stability	Iteration planning, management indicator of schedule convergence	SCOs opened vs. SCOs closed, by type (0,1,2,3,4), by release/component/ subsystem
Breakage and modularity	Convergence, software scrap, quality indicator	Reworked SLOC per change, by type (0,1,2,3,4), by release/compo-nent/subsystem
Rework and adaptability	Convergence, software rework, quality indicator	Average hours per change, by type (0,1,2,3,4), by release/component/ subsystem
MTBF and maturity	Test coverage/adequacy, robustness for use, quality indicator	Failure counts, test hours until failure, by release/component/ subsystem

MANAGEMENT INDICATORS

- There are three fundamental sets of management metrics: technical progress, financial status, and staffing
- progress. By examining these perspectives, management can generally assess whether a project is on budget and on schedule.
- Financial status is very well understood; it always has been.
- Most managers know their resource expenditures in terms of costs and schedule.

WORK AND PROGRESS

- The various activities of an iterative development project can be measured by defining a planned estimate of the work in an objective measure, then tracking progress (work completed over time) against that plan.
- Each major organizational team should have at least one primary progress perspective that it is measured against.

Default perspectives of this metric would be as follows:

- Software architecture team: use cases demonstrated
- Software development team: SLOC under baseline change management, SCOs closed
- Software assessment team: SCOs opened, test hours executed, evaluation criteria met
- Software management team: milestones completed

BUDGETED COST AND EXPENDITURES

- To maintain management control, measuring cost expenditures over the project life cycle is always necessary.
- Through the judicial use of the metrics for work and progress, a much more objective assessment of technical progress can be performed to compare with cost expenditures.
- Tracking financial progress usually takes on an organization-specific format.
- One common approach to financial performance measurement is use of an earned value system, which provides highly detailed cost and schedule insight.

Modern software processes are amenable to financial performance measurement through an earned value approach. The basic parameters of an earned value system, usually expressed in units of dollars, are as follows:

- Expenditure plan: the planned spending profile for a project over its planned schedule. For most software projects (and other labor-intensive projects), this profile generally tracks the staffing profile.
- Actual progress: the technical accomplishment relative to the planned progress underlying the spending profile. In a healthy project, the actual progress tracks planned progress closely.
- Actual cost: the actual spending profile for a project over its actual schedule. In a healthy project, this profile tracks the planned profile closely.

- Earned value: the value that represents the planned cost of the actual progress.
- Cost variance: the difference between the actual cost and the earned value. Positive values correspond to over-budget situations; negative values correspond to under-budget situations.
- Schedule variance: the difference between the planned cost and the earned value. Positive values correspond to behind-schedule situations; negative values correspond to ahead-of-schedule situations.

STAFFING AND TEAM DYNAMICS

- An iterative development should start with a small team until the risks in the requirements and architecture have been suitably resolved.
- Depending on the overlap of iterations and other project-specific circumstances, staffing can vary. For discrete, one-of-a-kind development efforts (such as building a corporate information system), the staffing profile in Figure 13-4 would be typical.
- It is reasonable to expect the maintenance team to be smaller than the development team for these sorts of developments.
- For a commercial product development, the sizes of the maintenance and development teams may be the same. When long-lived, continuously improved products are involved, maintenance is just continuous construction of new and better releases.

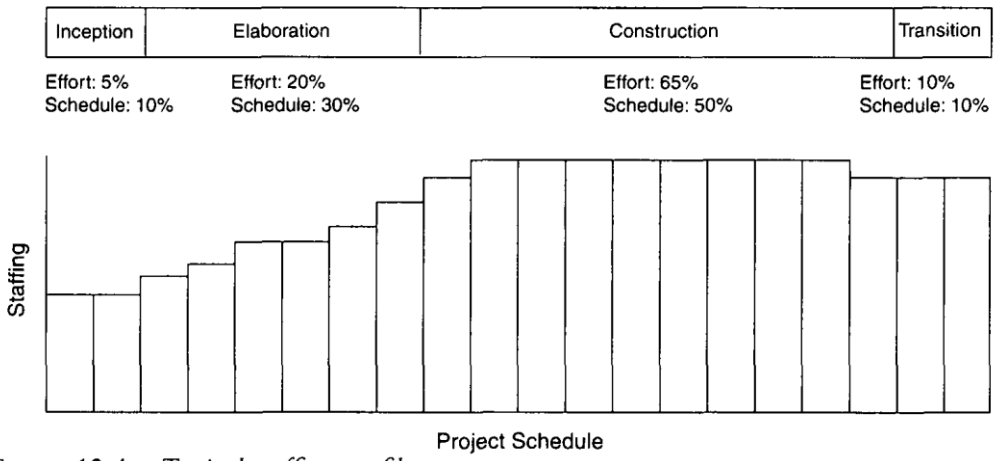


FIGURE 13-4. *Typical staffing profile*

QUALITY INDICATORS

- The four quality indicators are based primarily on the measurement of software change across evolving baselines of engineering data (such as design models and source code).

CHANGE TRAFFIC AND STABILITY

- Overall change traffic is one specific indicator of progress and quality.

- *Change traffic* is defined as the number of software change orders opened and closed over the life cycle (Figure 13-5).
- This metric can be collected by change type, by release, across all releases, by team, by components, by subsystem, and so forth.
- Coupled with the work and progress metrics, it provides insight into the stability of the software and its convergence toward stability (or divergence toward instability).
- *Stability* is defined as the relationship between opened versus closed SCOs.
- The change traffic relative to the release schedule provides insight into schedule predictability, which is the primary value of this metric and an indicator of how well the process is performing.
- The next three quality metrics focus more on the quality of the product.

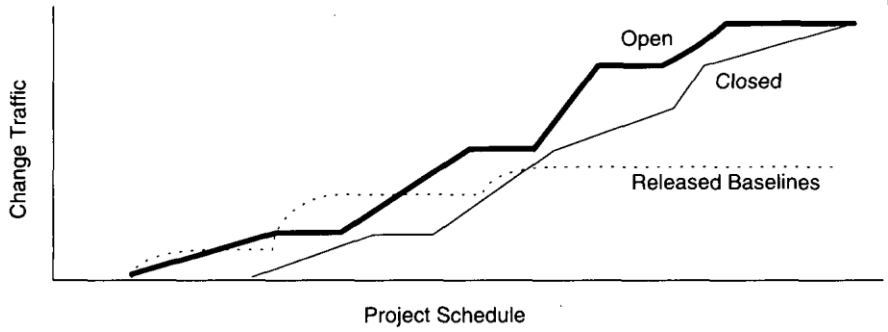


FIGURE 13-5. *Stability expectation over a healthy project's life cycle*

BREAKAGE AND MODULARITY

- *Breakage* is defined as the average extent of change, which is the amount of software baseline that needs rework (in SLOC, function points, components, subsystems, files, etc.).
- *Modularity* is the average breakage trend over time.
- For a healthy project, the trend expectation is decreasing or stable .
- This indicator provides insight into the benign or malignant character of software change.
- In a mature iterative development process, earlier changes are expected to result in more scrap than later changes. Breakage trends that are increasing with time clearly indicate that product maintainability is suspect.

REWORK AND ADAPTABILITY

- *Rework* is defined as the average cost of change, which is the effort to analyze, resolve, and retest all changes to software baselines.
- *Adaptability* is defined as the rework trend over time.
- For a healthy project, the trend expectation is decreasing or stable

MTBF AND MATURITY

- *MTBF* is the average usage time between software faults.

- In rough terms, MTBF is computed by dividing the test hours by the number of type 0 and type 1 SCOs.
- *Maturity* is defined as the MTBF trend over time (Figure 13-8).
- Early insight into maturity requires that an effective test infrastructure be established.
- Conventional testing approaches for monolithic software programs focused on achieving complete test coverage of every line of code, every branch, and so forth.

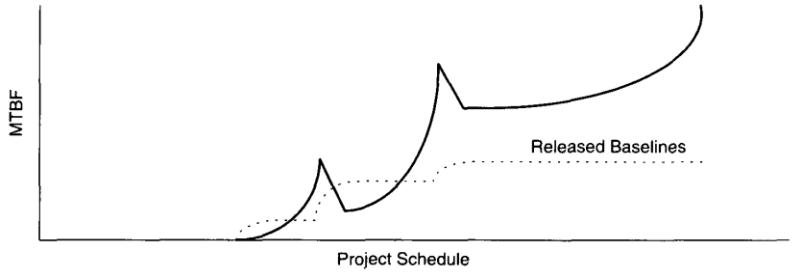


FIGURE 13-8. *Maturity expectation over a healthy project's life cycle*

LIFE-CYCLE EXPECTATIONS

There is no mathematical or formal derivation for using the seven core metrics. However, there were specific reasons for selecting them:

- The quality indicators are derived from the evolving product rather than from the artifacts.
- They provide insight into the waste generated by the process.
- Scrap and rework metrics are a standard measurement perspective of most manufacturing processes.
- They recognize the inherently dynamic nature of an iterative development process.
- Rather than focus on the value, they explicitly concentrate on the trends or changes with respect to time.
- The combination of insight from the current value and the current trend provides tangible indicators for management action.

The default pattern of life-cycle metrics evolution

METRIC	INCEPTION	ELABORATION	CONSTRUCTION	TRANSITION
Progress	5%	25%	90%	100%
Architecture	30%	90%	100%	100%
Applications	<5%	20%	85%	100%
Expenditures	Low	Moderate	High	High
Effort	5%	25%	90%	100%

Schedule	10%	40%	90%	100%
Staffing	Small team	Ramp up	Steady	Varying
Stability	Volatile	Moderate	Moderate	Stable
Architecture	Volatile	Moderate	Stable	Stable
Applications	Volatile	Volatile	Moderate	Stable
Modularity	50%-100%	25%-50%	<25%	5%-10%
Architecture	>50%	>50%	<15%	<5%
Applications	>80%	>80%	<25%	<10%
Adaptability	Varying	Varying	Benign	Benign
Architecture	Varying	Moderate	Benign	Benign
Applications	Varying	Varying	Moderate	Benign
Maturity	Prototype	Fragile	Usable	Robust
Architecture	Prototype	Usable	Robust	Robust
Applications	Prototype	Fragile	Usable	Robust

PRAGMATIC SOFTWARE METRICS

- Measuring is useful, but it doesn't do any thinking for the decision makers. It only provides data to help them ask-the right questions, understand the context, and make objective decisions.
- Because of the highly dynamic nature of software projects, these measures must be available at any time, tailorable to various subsets of the evolving product (release, version, component, class), and maintained so that trends can be assessed (first and second derivatives with respect to time).
- This situation has been achieved in practice only in projects where the metrics were maintained on-line as an automated by-product of the development/integration environment.

The basic characteristics of a good metric are as follows:

1. *It is considered meaningful by the customer, manager, and performer.* If any one of these stakeholders does not see the metric as meaningful, it will not be used. "The customer is always right" is a sales motto, not an engineering tenet. Customers come to software engineering providers because the providers are more expert than they are at developing and managing software. Customers will accept metrics that are demonstrated to be meaningful to the developer.

2. *It demonstrates quantifiable correlation between process perturbations and business performance.* The only real organizational goals and objectives are financial: cost reduction, revenue increase, and margin increase.

3. *It is objective and unambiguously defined.* Objectivity should translate into some form of numeric representation (such as numbers, percentages, ratios) as opposed to textual representations (such as excellent, good, fair, poor). Ambiguity is minimized through well-understood units of measurement (such as staff-month, SLOC, change, function point, class, scenario, requirement), which are surprisingly hard to define precisely in the software engineering world.

4. *It displays trends.* This is an important characteristic. Understanding the change in a metric's value with respect to time, subsequent projects, subsequent releases, and so forth is an extremely important perspective, especially for today's iterative development models. It is very rare that a given metric drives the appropriate action directly. More typically, a metric presents a perspective. It is up to the decision authority (manager, team, or other information processing entity) to interpret the metric and decide what action is necessary.

5. *It is a natural by-product of the process.* The metric does not introduce new artifacts or overhead activities; it is derived directly from the mainstream engineering and management workflows.

6. *It is supported by automation.* Experience has demonstrated that the most successful metrics are those that are collected and reported by automated tools, in part because software tools require rigorous definitions of the data they process.

METRICS AUTOMATION

- There are many opportunities to automate the project control activities of a software project.
- For managing against a plan, a software project control panel (SPCP) that maintains an on-line version of the status of evolving artifacts provides a key advantage.
- This concept was first recommended by the Airlie Software Council [Brown, 1996], using the metaphor of a project "dashboard."
- The idea is to provide a display panel that integrates data from multiple sources to show the current status of some aspect of the project.
- The panel can support standard features such as warning lights, thresholds, variable scales, digital formats, and analog formats to present an overview of the current situation.
- It can also provide extensive capability for detailed situation analysis.
- This automation support can improve management insight into progress and quality trends and improve the acceptance of metrics by the engineering team.

To implement a complete SPCP, it is necessary to define and develop the following:

- Metrics primitives; indicators, trends, comparisons, and progressions
- A graphical user interface: GUI support for a software project manager role and flexibility to support other roles
- Metrics collection agents: data extraction from the environment tools that maintain the engineering notations for the various artifact sets
- Metrics data management server: data management support for populating the metric displays of the GUI and storing the data extracted by the agents
- Metrics definitions: actual metrics presentations for requirements progress ^extracted from requirements set artifacts), design progress (extracted from design set artifacts),

implementation progress (extracted from implementation set artifacts), assessment progress (extracted from deployment set artifacts), and other progress dimensions (extracted from manual sources, financial management systems, management artifacts, etc.)

- Actors: typically, the monitor and the administrator

13.Tailoring the Process

Syllabus :

Tailoring the Process: Process discriminants.

13.1 Introduction

Tailoring the process involves the in-built activities required for the development task. Depending upon the project specific characteristics, the project framework is decided. Number of people included in the project team shows the scale of the project discrimination. In this chapter we will discuss about various key factors such as - stakeholder's relationships, process flexibility, maturity in the process, and architectural risks.

13.2 Process Discrimination

There are two different dimensions of the project Discrimination,

1. Technical Dimensions
2. Management Dimensions

Both of these dimensions can be measured from its lower to higher complexity variability. Following table describes the differentiation between lower and higher technical/management complexity. That is, indirectly, it describes the factors which decide that whether the dimension has lower technical/management complexity or higher technical/management complexity.

Higher Technical Complexity	Lower Technical Complexity
Embedded, real time, distributed, fault tolerant	Clear-cut Automation
High performance	Interactive performance
Portable and hence might be multiple platform	Single platform
Unprecedented	Many precedent system
Architectural re – engineering	Application re-engineering

Higher Management Complexity	Lower Management Complexity
High Scale – More number of peoples work in the team	Low scale – limited peoples (Smaller team) works in the team
Large stakeholders	Limited stakeholders
The final output is "Project"	The final output is "Product"

The average software project which is neither lower nor higher contains following attributes;

- Project development team contains 5 to 12 members
- Completion time of project is about for 10 - 12 months
- External interfaces range from 5 - 6
- Existence of few Minor risks

13.2.1 Scale

There are so many ways to measure the scale of the software application, some of them are mentioned below;

1. Total number of lines in the source code
2. Exact figure of function point
3. Total use cases
4. Total dollars

In case of the process tailoring, the scale is measured by the *total num ber o f the people involved in the process as a team.* There are different team sizes available which work together for the process tailoring. People involved in the team have specific names as follows;

1 individual Trivial

5 member's team -> Small

25 member's team -> Moderate

125 member's team Large

625 member's team -> Huge and so on Within a single organization there are various personal managements involved in case if the team members increase or decrease. There is lot of work related to

the management, such as planning for the process, administration work, coordination among all the team members, communication, progress assessment, review, administration etc. Now we discuss about the team size and their management duties;

1. Trivial Project

There is no management overhead since single individual handle all the management related work.

- For the intermediate artifact little most documentary required.
- Single threaded workflow.
- Individual personal skill shows only the performance

2. Small Team Project

- Little management overhead
- Communication regarding the artifacts done between team members
- Objective for the team members is common
- Planning done regarding the milestone or checkpoints of the project. These planning are informally conducted and changeable.
- Single workflow
- As small size of team performance also depends on the personal skills of the team members. Individual performance has more weightage, hence it is countable.

3. Moderate Team Project

- Moderate management overhead, this overhead is about assessment, review and coordination. All this is done by dedicated software project manager within the team. They are communicating, coordinate and assess the project work and workflows. And one more major duty they make resource balancing.
- Compulsory communication is required for the artifacts between team member Milestones and checkpoints are somewhat planned and conducted
- Small workflow exiting between team members
- The overall performance about the project totally depends mainly upon the personal leaders.

4. Large Team Project

- As large project, large management overhead
- Dedicated software project managers are there, but for the supporting kind of view other subproject managers are also available. Again, they handle workflow and make balance resourcing.
- Communication between different team members are essential
- Large concurrent workflow existing between team members.
- Performance of the project depends upon the software project manager, subproject manager and team leaders.

5. Huge Project

- Substantial management overhead, including many software project manager and subproject managers.

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

- Their primary duty is handle workflow and balance resourcing
- Communication engineering produce results in the team members
- Large concurrent workflow existing between team members.
- Performance of the project depends upon the software project manager,
- subproject manager and team leaders.

13.2.2 Stakeholder Cohesion and Contention

There are several stakeholders such as developers, maintainers, buyers, contractors, sub-contractors, users etc. exists in the process discrimination for the cooperation and coordination. Cohesive teams are having particular settle goals, complementary skills and close communication. To achieve primary focus goal that is 'profit' sole organization can utilized for fund, developed, marketed and finally sold the product. Excellent communication is done between team members to increase the fruitfulness of the product establishment. A small cohesive skilled based organization is set up for that purpose. ***Development Contractor*** and ***funding authority*** are the two different aspects regarding the software project completion just as funding authority thinks for;

- Minimize cost
- Maximize feature set
- Speed up time to market

Even ***users*** of the product also think same like funding authority. But development contractor thinks only about *maximizing the profitability*.

13.2.3 Process Flexibility or Rigor

At the time of project process, a specific project contract exists which includes;

- *Visionary document* : This kind of document includes the overall objective regarding the project. The goal covered and targeted for the project also supports the vision document.
- *Development Plan* : Project evaluation in some specific path by which its development done. How project develop? How much duration it takes? What kind of artifact used in it? All such development planning is included.
- *Business case* : How the project use? What are the rules and regulation? Actually, rigor and flexibility are two opposite things and both are mentioned in the project's contract.

In case of flexible or sophisticated or loose contacts, then in such cases;

- Management complexity is minimal
- All the development processes should be changed easily
- Feature set, time to market, quality and budget for the project also to be changed. All this is freely traded.
- Less and little overhead
- Less time to utilize new changes.

302 PARANJPE UDYOG BHAVAN, OPP MAHESH TUTORIALS, THANE STATION,THANE WEST, 8097071144

- Coordination provided from the managers like development manager, business unit handle managers, marketing managers.

For example; if a certain organization wants to change the designing part which has more budget then the releasing order for the same is done in the same week. Opposite to that in rigor contract the opposite situations are there.

- High management complexity
- Smallest change in the development processes not done very easily
- High overhead
- Many months (No fixed number to say) to utilize few changes.

13.2.4 Process Maturity

There are various mature and immature processes. Simplicity exists in mature process. This is not the same with immature process. Clear-cut task in certain processes use tailored process which is a mature process in the organization.

13.2.5 Architectural Risk

Many architectural risks arise in the project sources. Some of the architectural risks are mentioned below;

Regarding system performance:

- Accuracy provided by the system
- Resources utilization
- Response time
- Throughput Regarding the Change in Robustness
- New addition of artifact
- New feature addition
- New dynamic operation conditions addition
- New technology evaluation Regarding reliability in the system
- Feature predictable habits
- Fault acceptance

Review Questions

Q. 1 What is meant by 'tailoring the process'.
Q. 2 Differentiate between higher and lower technical complexity.
Q. 3 Differentiate between higher and lower management complexity.
Q. 4 Describe Stakeholder Cohesion and Contention.
Q. 5 List out the architectural risks.
Q. 6 Define process flexibility / rigor.
Q. 7 Describe the measures on which the scale is measured.
Q. 8 Describe the management duties of the team size.

UNIT VI

14.Future Software Project Management

Syllabus :

Future Software Project Management: Modern Project Profiles, Next generation Software economics, modern process transitions.

14.1 Continuous Integration

At begin of the life cycle of the iterative development, primarily architecture done, which allow the verification activity within design phase. This concept avoids the big bang produce in the continuous integration. If we consider the conventional software management, then it follows the late design breakage. This is because it posses the sequential traditional activities for the software project management which are 1) Requirements, 2) Design, 3) Coding, 4) Integration and finally 5) Testing.

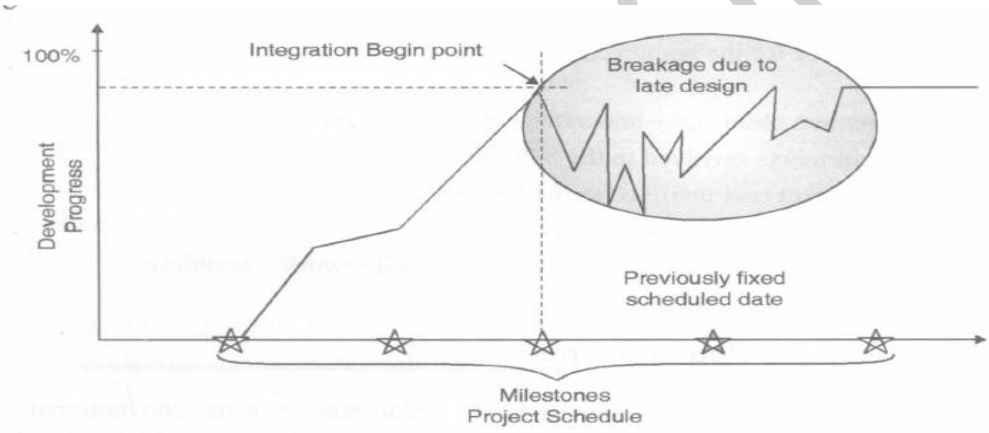


Fig. 14.1.1 : Conventional Software Project Management

This issue solved in the modem projects, which covers the 100% software project management at the previously scheduled date. At the design phase architecture first approach forces integration which includes the demonstration construction level. This beginning work gives fast track for the future work. Design breakage covered in the engineering phase.

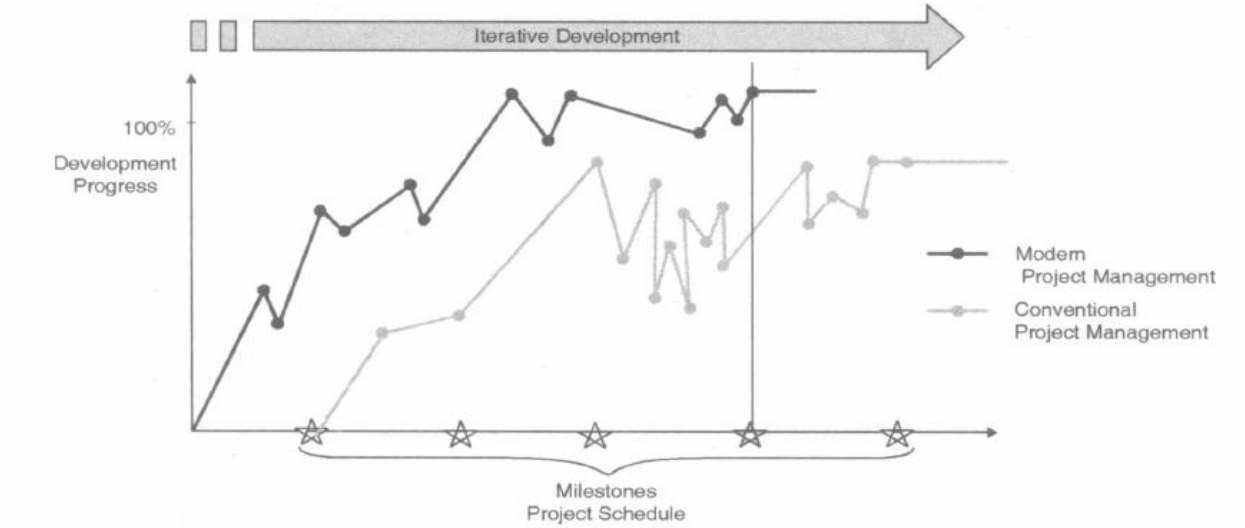


Fig. 14.1.2 : Modern Project Management verses conventional project Management

14.2 Early Risk Resolution

As we know that the risks occurrences and its resolution had done in the engineering stage of the life cycle which is means inception and elaboration stage. That indicates production phase free for the resource commitment. All this is possible due to architecture first principle. The risk management philosophy understands by following points;

- 20% requirements involved in the 80% engineering
- 20% component cost involved in the 80% software cost
- 20% components posses 80% errors
- 20% changes follows the 80% software scrap and rework scheduling
- 20% peoples makes 80% progress

14.3 Evolutionary Requirements

The unit requirements are smallest kind of requirement in the conventional approach, which completes further components requirements. Components requirements fulfillment for the sub system and finally sub system requirements finishes the system requirements.

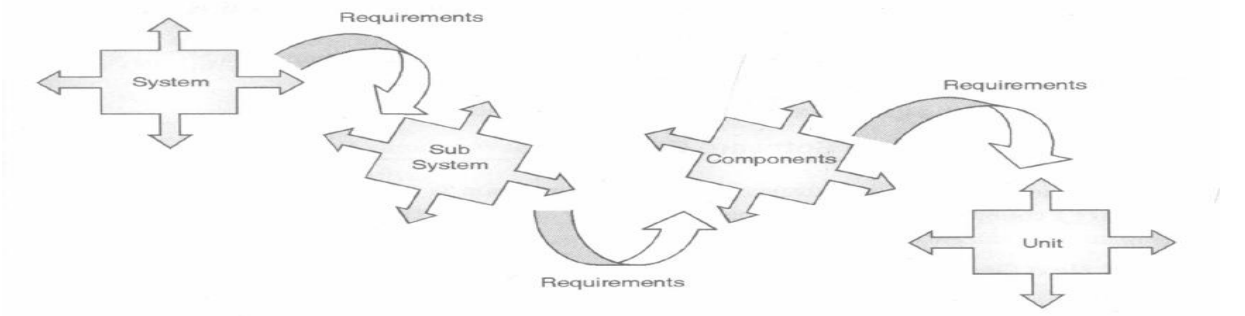


Fig. 14.3.1 : Requirements Decomposition in conventional approach

14.4 Teamwork among Stakeholders

Good and healthy relationship between stakeholders makes various strong activities for software project. Stakeholders are customer, users, monitors, contractors, software engineers etc. All objectives of the software are understood by the all three stakeholders such as monitor, users or customer. If and only if they are software experts then they are applicable for process handling. The best organization always keep in mind that "Customer is King", and hence they always try how he is satisfy by the product produce by the firm. Here customer want quality software and organization will go for profitability.

14.5 Top - Ten Management Principles

Following are the top ten management principles on the modern software profile is base :

1. Architecture first is the primary base of the process.
2. Earlier risk finding is good strategy for process which is the part of the iterative life cycle.
3. Component based development requires the transition design.
4. Generate change management profile projects.
5. In case of the round trip engineering improve change freedom activity.
6. In the model design notations draw the designs artifacts.
7. Process instrumentation is necessary for the quality control objectives and assessment of progress in the project.
8. Demonstration based approach is useful for assessing middleware artifacts.
9. Detailing regarding each level is at releasing time.
10. Generate configuration process which is economic scalable.

14.6 Next Generation Software Economics

Practice made by the advance software organization is nothing but **Next Generation Software Economics.** This returns the return policy against investment. We can say that this is the mature organization indication.

14.6.1 Next Generation Cost Models

Software expertise team introduces different - different software cost estimation models which are produce different reason for estimation of it. Some of them are listed below;

1. Old C++ verses GUI based Java
2. Function oriented verses code of source lines
3. Scale of economy verses scale of diseconomy
4. Quality against productivity measure
5. Procedure oriented verses object oriented
6. New Profitable verses convention traditional development As today Architecture comes First software models use, vendors of it focuses on the up to date project data which are based on the modem software project profile. The cost of architecture baseline is depends upon the

function of scale, quality they maintain, technology they usage, process they work and teamwork they achieve. Architecture baseline includes designing, producing, testing and maintaining.

14.6.2 Modern Software Economics

Following are the changes should made by the organizational Manager;

1. Software problems should be fixed earlier in the design phase is better. If we find out it after delivery cost increases up to 100%.
2. Schedule which is created for software development is reducing up to only 25% but not more than that.
3. Maintenance is so much important in side of the customer. Note that if use some rupees for development maintenance takes twice of that.
4. Total lines of the source code of the program generates the software development and maintenance cost.
5. Productivity depends upon the team of the various people.
6. It is observed that, Software: Hardware ratio should be increase in every year.
7. 15% software development is because of the programming.
8. As per SLOC software and product cost is 3 times higher than normal single software
9. problem.
10. 60% errors are catch through Walkthroughs.
11. 20% contributors involves in 80% contributions.

14.7 Modern Process Transition

If and only if there is hard work then it gives successful software management. There are some new fundamental concepts add in the modem process such as Technologies along with various features Opportunities involve innovations Dimensions along with complexities. Automation Avenues Extra priorities for the new customers

14.7.1 Culture Shifts

Culture shifts are design for the some of the adjustments.

- Performers are major indicators which are lower and middle level managers.
- Customer given requirements and its design should be change and tangible.
- Motivated demonstrations are encouraged.
- Performance of the project which is either bad or good should be clear earlier in the life cycle.
- If increments are shown earlier they are immature.
- Uses of artifacts are more in later stage than early stage.
- Whatever problems arises that are faced and solved step by step
- The teams who are working for the software take responsibility of quality maintenance. There is no separate unit or department for it.

- Life cycle of the software model has issue of the performance of the software.
- Automation investigation is necessary.
- Profits are arises only from those organization which are good in service.

Review Questions

Q. 1 Why modern project management is better than conventional project management ?

Q. 2 Explain 20 : 80 pattern in early risk resolution.

Q. 3 What do you mean by Evolutionary requirements.

Q. 4 Why teamwork among stack holder is essential ?

Q. 5 What are top ten management principles ?

Q. 6 Explain next generation cost model.