

Unit -1 SOFTWARE PROBLEM & SOFTWARE PROCESS

1.1 Cost, Schedule, and Quality:

- In the industrial-strength software domain, there are three basic requirements - cost, schedule, and quality.
- The software should be produced at reasonable cost, in a reasonable time, and should be of good quality.
- These three parameters defines a software project.

COST:

- Industrial-strength software is very expensive mainly because of high labor.
- Lines of code (LOC) or thousands of lines of code (KLOC) is commonly used to measure of software size in the industry.
- The cost of developing software is usually measured in terms of person-months of effort.
- Productivity is frequently measured in the industry in terms of LOC (or KLOC) per person-month.
- This productivity depends on the entire development cycle, not just the coding task.
- Software companies of ten charge the client between \$3000 - \$15,000 per person-month.
- With a productivity of 1000 LOC per person-month, even small projects can easily end up with software of 50,000 LOC and the software project will cost between \$150,000 and \$750,000.

SCHEDULE:

- Schedule is another important factor in many projects.
- Time to market of a product should be reduced. (Cycle time from concept to delivery should be small)
- It means software needs to be developed faster, and within the specified time.
- Reducing the cost and the cycle time for software development are central goals of software engineering.

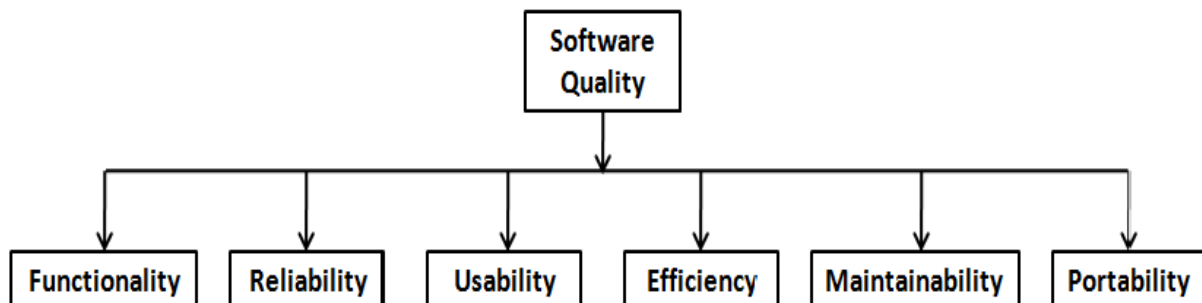
- If productivity is higher, it should be clear that the cost in terms of person-months will be lower .
- A team of higher productivity will finish the job in less time than lower productivity team.
- Hence higher productivity is a basic driving force.

QUALITY:

- Quality is one of the key features required while developing software.
- But many software's developed have problems regarding unreliability.
- The software often does not do what it is supposed to do or does something it is not supposed to do.
- Hence developing high-quality software is another fundamental goal of software engineering.

Software quality comprises six main attributes such as:

- 1) **Functionality:** The capability to provide functions which meets needs when the software is used.
- 2) **Reliability:** The capability to provide failure-free service.
- 3) **Usability:** The capability to be understood, learned, and used.
- 4) **Efficiency:** Capability to provide necessary performance relative to the amount of resources used.
- 5) **Maintainability:** Capability to be modified for purposes of making corrections, improvements, or adaptation.
- 6) **Portability:** The capability to be adapted



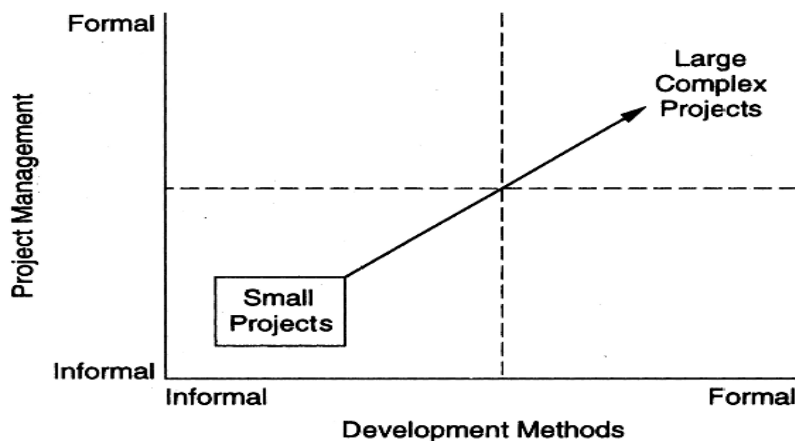
- Different projects concentrate on different attributes & reliability is the main quality criterion.
- Unreliability of software is due to the presence of defects in the software.

- Quality objective is to reduce the number of defects as much as possible.
- To determine the quality of a software product, we need to determine the number of defects in the software that was delivered is to log the defects during development and after delivery.
- Such defects must be defined clearly (System crash, word misspells) and appropriate actions should be taken to remove those defects.
- Another quality attribute is *maintainability*.
- Once the software is delivered and deployed, it enters the maintenance phase.
- *Corrective maintenance* is where defects discovered during maintenance phase will to be removed.
- Maintenance is also needed to change the delivered software to satisfy the enhanced needs of the users and the environment, leading to *adaptive maintenance*.

1.2 Scale and Change:

- Two more characteristics of the problem domain that also influence the solutions are scale and change.
- Most industrial-strength software systems are large and complex, requiring tens of thousands of LOC.
- For example Windows XP software has 40,000 KLOC and open ssl has 200KLOC.
- Developing a large system requires a different set of methods compared to developing a small system.
- For example take the problem of counting people in a room versus taking a census of a country.
- Census will require considerably more management, organization & validation, in addition to counting.
- Similarly, methods used to develop programs of smaller size may not to work for larger sized software.
- A different set of methods must be used for developing large software.
- Any software project involves the use of engineering and project management.
- In small projects, informal methods for development and management can be used.
- However, for large projects a proper method for engineering the system has to be employed and the project has to be tightly managed to make sure that cost, schedule, and quality are under control.

- Change is another characteristic of the problem domain.



- The complete set of requirements for the system is generally not specified at the beginning of the project.
- As development proceeds and time passes, additional requirements are identified.
- This need for changes requires that methods should allow change and should work efficiently.
- Change requests need to be handled carefully else it can be quite disruptive to a project.
- Overall, changes in requirements should be accepted and accommodated.

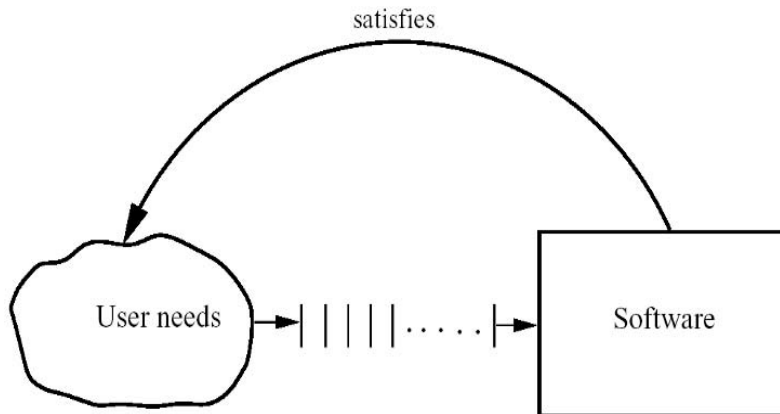
1.3 Software Processes:

“Software engineering is defined as the systematic approach to the development, operation, maintenance, and retirement of software.”

- During software delivery, high quality, low cost, and low cycle time are important goals which software engineering must achieve.
- The system approach must help achieve a high quality and productivity (Q&P).
- In software, the three main factors that influence Q&P are people, processes, and technology.
- Tools are aids that help people perform some of the tasks more efficiently and with fewer errors.
- Processes form the heart of software engineering, with tools and technology providing support.

1.4 Process and Project:

- A process is a sequence of steps performed for a given purpose.

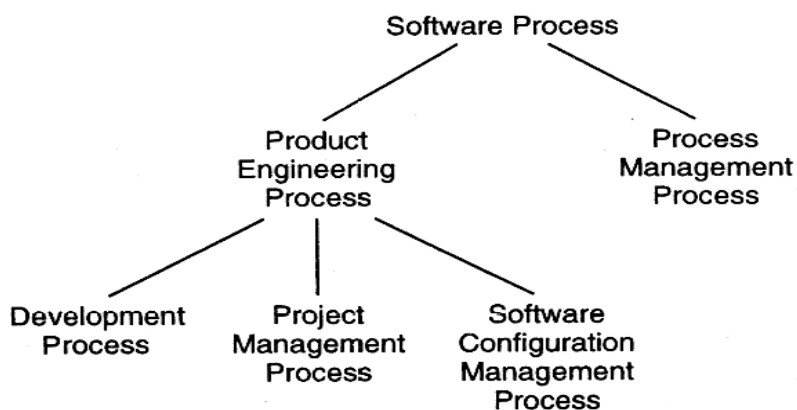


- The purpose of developing industrial strength software is to satisfy the needs of users or clients.
- A software project is one instance of this problem, and the development process achieves the purpose.
- In a project, a process specification may be used as the process the project plans to follow.
- The actual process is what is actually done in the project.

1.5 Component Software Processes:

“The processes that deal with the technical and management issues of software development are collectively called the software process.”

- A software project will have to engineer a solution and properly manage the project.



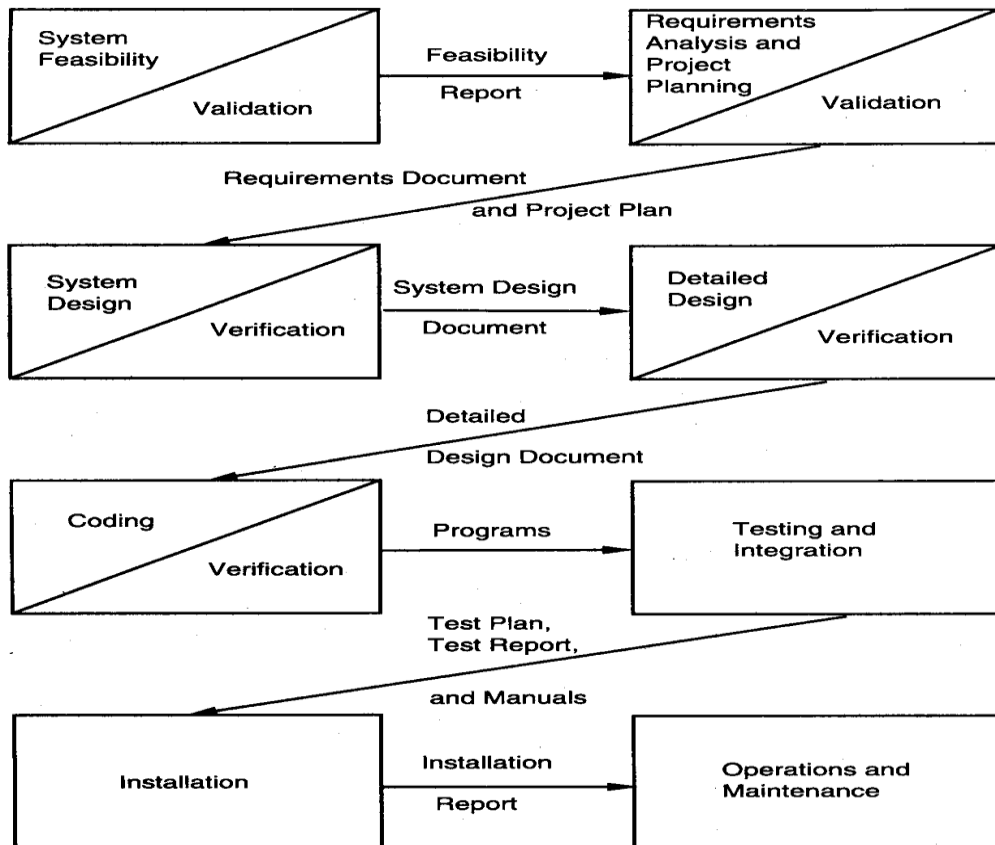
- There are two major components in a software process - a ***development process*** and a ***project management process***.
- The ***development process*** specifies all the engineering activities that need to be performed.
- The ***management process*** specifies how to plan and control these activities so that cost, schedule, quality, and other objectives are met.
- Development processes generally do not focus on evolution and changes, to handle them another process called ***software configuration control process*** is used.
- The three processes focus on the projects & products comprising the ***product engineering processes***.
- A software process is a dynamic, as it must change to adapt to newer technologies and tools.
- The whole process of understanding the current process, analyzing, determining how to improve, and then implementing that improvement is handled by the ***process management process***.
- The basic objective of the process management process is lower cost and improving quality.
- In a typical project, development activities are performed by ***programmers, designers, testers***, etc.
- The project management process activities are performed by the ***project management***.
- Configuration control process activities are performed by a group called the ***configuration controller***.
- Process management process activities are performed by ***software engineering process group (SEPG)***.

1.6 Software Development Process Models:

- The goal of software development process is to produce a high-quality software product.
- It focuses on activities directly related to production of the software, For ex: design, coding, and testing.
- Let us discuss some of the major models.

1.6.1 Waterfall Model:

- The simplest process model is the waterfall model.
- The different phases in the waterfall model are organized in a linear order.
- The model was originally proposed by Royce, Nowadays many variations of the model have evolved.
- The model was originally proposed by Royce, Nowadays many variations of the model have evolved.



THE WATERFALL MODEL

- In this model, a project begins with feasibility analysis.
- Next the requirements analysis and project planning begins.
- The design starts after the requirements analysis is complete, and later on coding begins.
- Once the programming is completed, the code is integrated and testing is done.
- Upon successful completion of testing, the system is installed.

- After this, the regular operation and maintenance of the system takes place.
- Each phase deals with a distinct and separate set of concerns (problems).
- By doing this, the large and complex task is broken into smaller tasks.
- Verification and validation ensures that the output of each and every phase is consistent.
- Documents are produced in each project: Requirements document, Project plan, Design documents, Test plan and test reports, Final code, Software manuals (ex: user, installation, etc.)

Advantages:

- One of the main advantages of the waterfall model is its simplicity.
- It is straightforward and divides the large task of building a software system into phases.
- It is also easy to administer in a setup

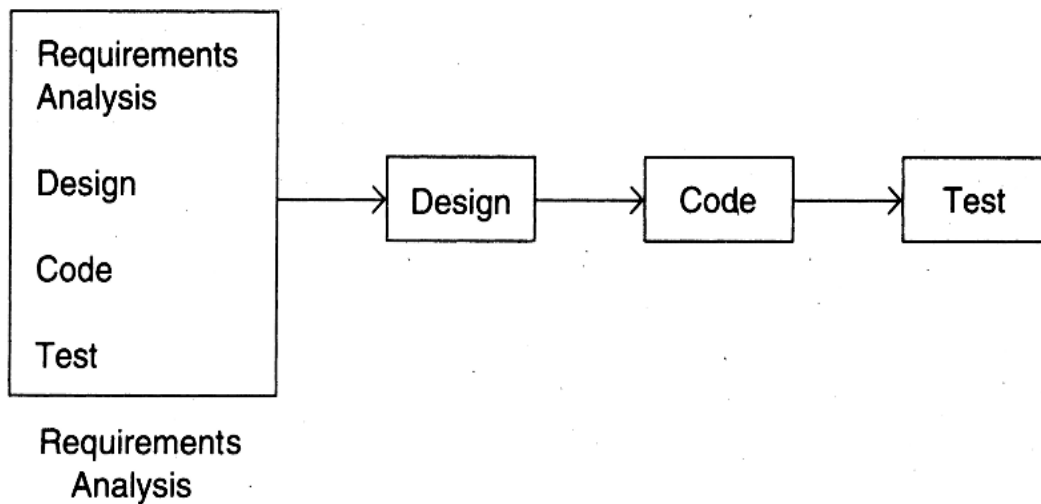
Some of the key limitations are:

- Requirements of a system are fixed, user may not know the requirements at the beginning itself.
- Fixing the requirements includes choosing the hardware and if a large project takes few years to complete. Then the final software will use older version hardware technology which is not desirable for such expensive software systems
- The entire software is delivered in one shot at the end. This involves heavy risks, as the user does not know until the end what they are getting & if the project runs out of money in the middle, then there will be no software.
- All requirements must be specified at the start, it encourages the users to add features which they think might be needed (which finally may not get used).
- It is a document-driven process that requires formal documents at the end of each phase.
- Despite these limitations, the waterfall model has been the most widely used process model.
- If requirements for the software are quite clear, the waterfall model works well, and may be the most efficient process.
- Despite these limitations, the waterfall model has been the most widely used process model.

- If requirements for the software are quite clear, the waterfall model works well, and may be the most efficient process.

1.6.2 Prototyping:

- The goal of a prototyping-based development process is to overcome the limitation of waterfall model.
- Instead of fixing the requirements before any design or coding can proceed, a throwaway prototype is built to help understand the requirements.
- This prototype is developed based on the currently known requirements.
- Development of the prototype undergoes design, coding, and testing in an informal way.
- After the prototype is developed, the clients provide feedback to the developers regarding the prototype: what is correct, what needs to be modified, what is missing, what is not needed, etc.
- By using this prototype, client can get a better understanding about the requirements of the system.
- Based on the feedback, the prototype is modified to include some of the suggested changes that can be done easily, and then the users and the clients are again allowed to use the system.
- This cycle repeats until clients are satisfied with the software.
- This results in more stable requirements that change less frequently.
- Prototyping is an attractive idea for complicated and large systems.
- Letting the client interact with the prototype helps in determining the actual requirements for the system.
- Requirement analysis for the prototype must be feasible, its cost must be kept low.
- Exception handling, recovery, some standards and formats are usually not included in prototypes.
- Because the prototype is to be thrown away, only minimal documentation is done during prototyping.
- Another cost-cutting measure is to reduce testing as it is a major part of development expenditure

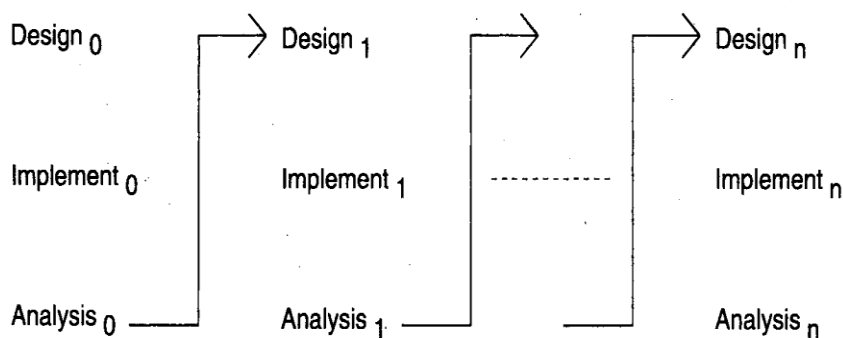


The prototyping model

- The experience of developing the prototype will reduce the cost of the actual software development.
- The quality of final software is likely to be far superior, as the experience engineers have obtained while developing the prototype will enable them to create a better design, write better code, and do better testing.
- Overall, prototyping is well suited for projects where requirements are not clear.

1.6.3 Iterative Development:

- The iterative development process model overcomes the limitations of the waterfall model.
- It combines the benefits of both prototyping and the waterfall model.
- The basic idea is that the software should be developed in increments.



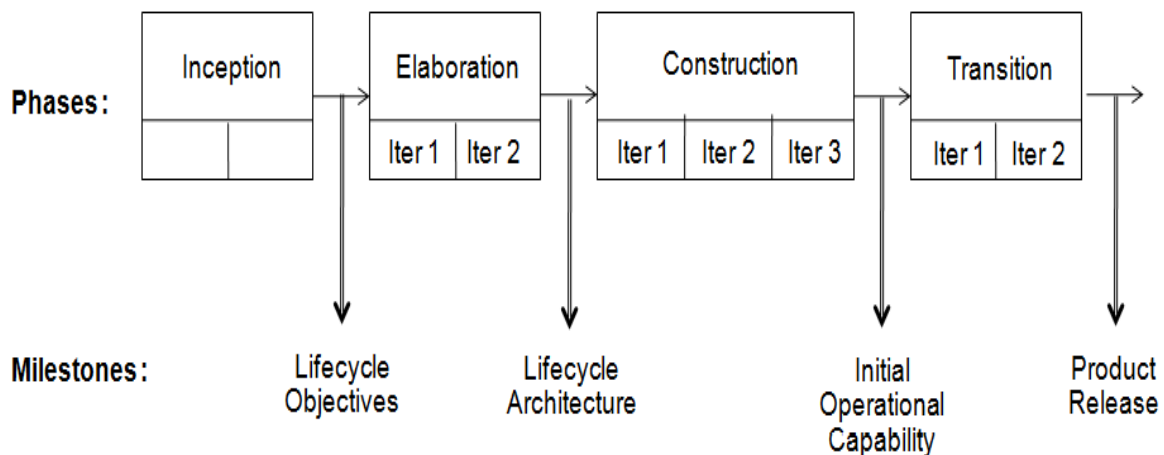
The iterative enhancement model.

- Each increment adding some functional capability to the system until the full system is implemented.
- In the first step of this model, a simple initial implementation is done.
- A ***project control list*** is created that contains, all the tasks that must be performed to obtain the final implementation.
- This project control list gives an idea about the progress of the project.
- Each step consists of removing the next task from the list, designing the implementation for the selected task, coding and testing the implementation, performing an analysis and updating the list as a result of the analysis.
- These three phases are called the design phase, implementation phase, and analysis phase.
- Based on the analysis, one of the tasks in the list can include redesign of defective components or redesign of the entire system.
- The process is iterated until the project control list is empty.
- Each entry in the list is a task that should be performed in one step of the iterative enhancement process.
- It should be simple enough to be completely understood.
- Selecting tasks in this manner will minimize the chances of error and reduce the redesign work.
- The design and implementation phases of each step can be performed in a top-down manner or by using some other technique.
- Iterative method can be quite expensive due to changing requirements.
- Changes in future iterations may lead to extra rework and/or discarding of work done earlier.
- Overall, it may not offer the best technical solution, but there are more benefits.
- Another common approach for iterative development is to apply standard waterfall or prototyping approach (collecting requirements), but deliver the software iteratively.

- It was designed for object-oriented development using the Unified Modeling Language (UML).
- Here the development of software is divided into cycles, each cycle delivering a fully working system.
- Each cycle is executed as a separate project whose goal is to deliver some additional capability to an existing system (built by the previous cycle).
- Each cycle itself is broken into four consecutive phases:

1. Inception phase 2. Elaboration phase 3. Construction phase 4. Transition phase.

- Completion of each phase provides some clearly defined outputs.
- Purpose of *inception phase* is to establish the goals and scope of the project and completion of this phase is the *lifecycle objectives* milestone.
- This phase specifies the vision and high-level capability of the future system, benefits, key use cases, major risks, and a basic plan of the project regarding the cost and schedule.
- Based on the output of this phase, a go/no-go decision may be taken by the stakeholders.
- In *elaboration phase*, the architecture of the system is designed, based on the detailed requirements analysis. The completion of this phase is the *lifecycle architecture* milestone.
- By the end of this phase, most of the requirements, the choice of technologies, architecture, etc. have been taken, and a detailed understanding of the project exists.
- Outputs of this milestone allow technical evaluation of the proposed solution, as well as a better informed decision about cost-benefit analysis of the project.
- In the *construction phase*, the software is built and tested. Completion of this phase achieves the *initial operational capability* milestone.



The RUP model

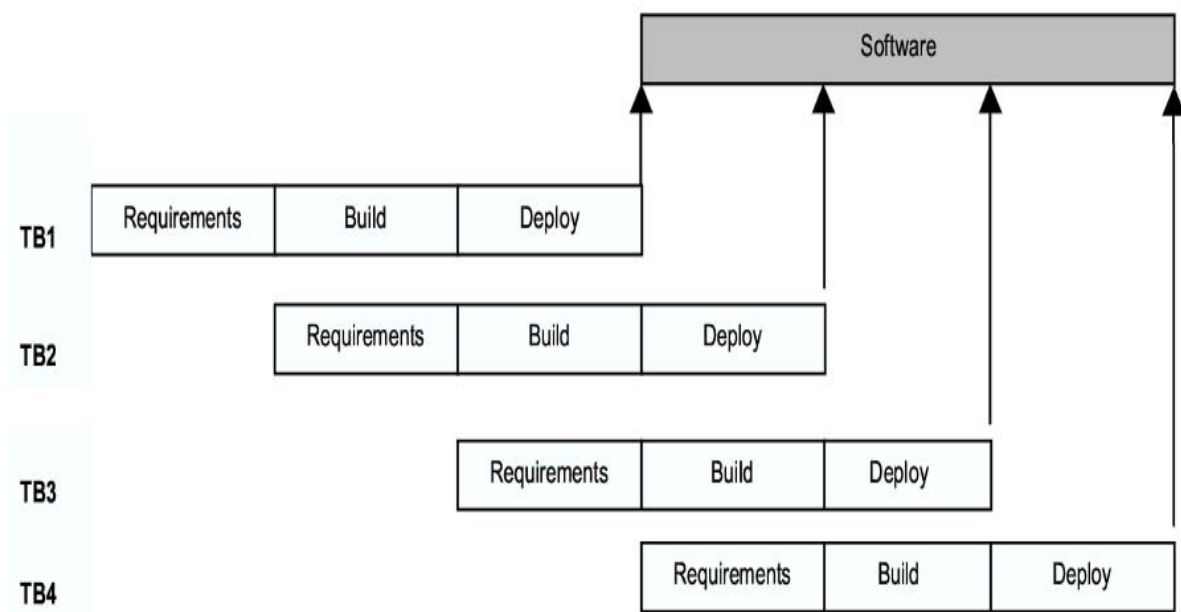
- This phase results in the software product to be delivered, along with associated user and other manuals.
- The purpose of the **transition phase** is to move the software from the development environment to the client's environment, where it is to be hosted (installed).
- This is a complex task which can require additional testing, conversion of old data for this software to work, training of personnel, etc.
- The successful execution of this phase results in achieving the milestone **product release**.
- Generally the construction phase will be broken into multiple iterations, each iteration producing a working system which can be used for feedback, evaluation etc.
- Different engineering activities maybe performed in a phase to achieve its milestones.
- RUP groups the activities into different sub processes which it calls **core process workflows**.
- These sub processes correspond to the tasks of performing requirements analysis, doing design, implementing the design, testing, project management, etc.
- One key difference of RUP from other models is that it has separated the phases from the tasks and allows multiple of these sub processes to function within a phase.
- RUP allows requirement activity even in construction phase, which was not possible in waterfall model.
- Sub process can be active in many phases but the effort spent on the sub process will change with phases.
- For ex: a lot more effort will be spent in the requirement sub process during elaboration, and less time during construction, and still less in transition.

- Overall, RUP provides a flexible process model, and also captures the reality of the situation as it is not possible to specify all requirements at the start.

1.6.5 Timeboxing Model:

- To speed up development, parallelism between the different iterations can be employed.
- A new iteration begins before the system produced by the current iteration is released, and hence development of a new release happens in parallel with the development of the current release.
- By starting an iteration before the previous iteration has completed, it is possible to reduce the average delivery time for iterations.
- In the timeboxing model, the basic unit of development is a time box, which is of fixed duration.
- Since the duration is fixed, requirements should be able to fit into the time box.
- This is in contrast to regular iterative approaches where the functionality is selected and then the time to deliver is determined.
- Timeboxing makes the schedule a nonnegotiable and a high-priority commitment.
- Each time box is divided into a sequence of stages, like in the waterfall model.
- Each stage performs some clearly defined task for the iteration and produces a clearly defined output.
- A dedicated team is selected for each stage where one team performs only tasks of that stage and other stages are performed by others teams.
- Having time-boxed iterations with stages of equal duration and having dedicated teams allows different iterations to work in parallel.
- To illustrate the use of this model, consider a time box consisting of three stages: requirement specification, build, and deployment.
- The requirement stage is executed by its team of analysts and specifies a list of requirements to be built in this iteration along with a high-level design.
- The build team develops the code for implementing the requirements, and performs the testing.
- The tested code is then handed over to the deployment team, which performs pre-deployment tests, and then installs the system for production use.

- These three stages are such that they can be done in approximately equal time in an iteration. With a time box of three stages, the project proceeds as follows.
- When the requirements team has finished requirements for timebox-1, the requirements are given to the build team for building the software.
- The requirements team then goes on and starts preparing the requirements for timebox-2.
- When the build for timebox-1 is completed, the code is handed over to the deployment team, and the build team moves on to build code for requirements for timebox2, and the requirements team moves on to doing requirements for timebox-3.



Executing the timeboxing process model.

- For example, if the time box duration T is 9 weeks (each stage duration is 3 weeks), the first delivery is made 9 weeks after the start of the project.
- The second delivery is made after 12 weeks, the third after 15 weeks, and so on.

Requirements Team	Requirements Analysis for TB1	Requirements Analysis for TB2	Requirements Analysis for TB3	Requirements Analysis for TB4
Build Team		Build for TB1	Build for TB2	Build for TB3
Deployment Team			Deployment for TB1	Deployment for TB2

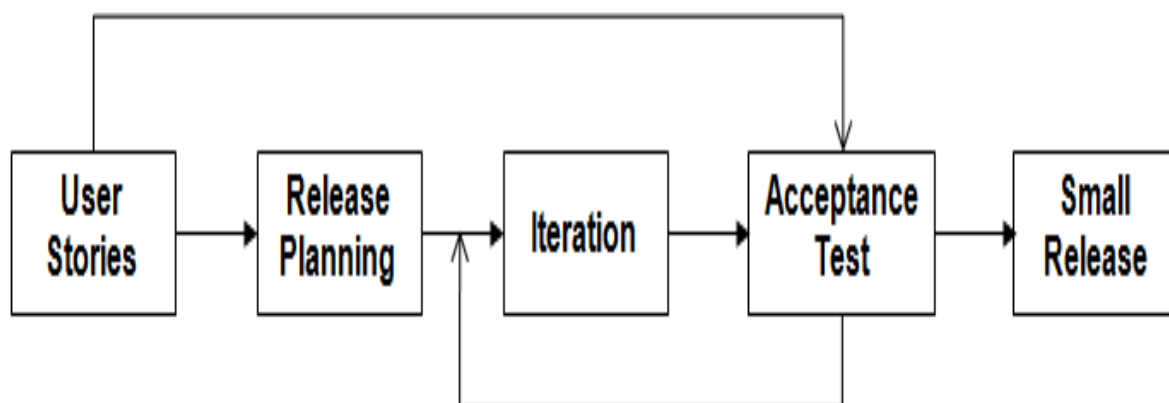
Tasks of different teams.

- It should be clear that the duration of each iteration has not been reduced.
- The total work done in a time box and the effort spent in it also remains the same.
- The real cost of this reduced time is in the resources used in this model.
- Time boxing provides an approach for utilizing additional manpower to reduce the delivery time.
- Time boxing is well suited for projects that require a large number of features to be developed in a short time using stable technologies.

1.6.6 Extreme Programming and Agile Processes:

- Agile approaches evolved in the 1990s to overcome the problem of documentation based processes, particularly the waterfall approach.
- Agile approaches are based on some common principles, some of which are :
 - 1) Working software is the key in a project.
 - 2) Software should be developed and delivered quickly in small increments.
 - 3) Even late changes in the requirements should be allowed.
 - 4) Face-to-face communication is preferred over documentation.
 - 5) Continuous feedback and involvement of client is necessary for developing good-quality software.
 - 6) Simple design which evolves and improves with time is the better approach .

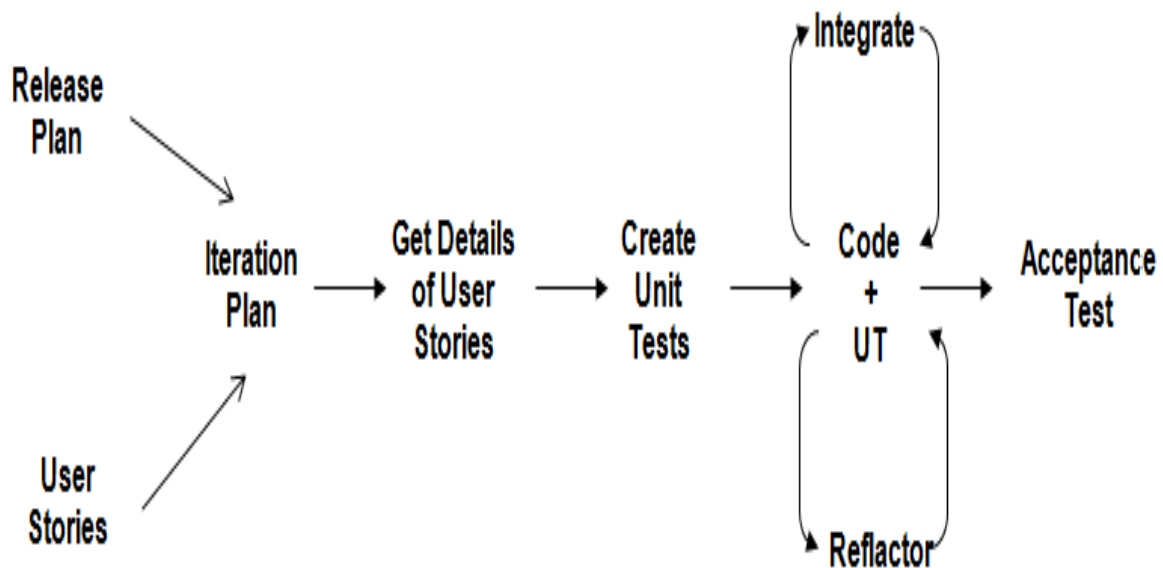
- 7) The delivery dates are decided by experts.
- 8) Many detailed agile methodologies have been proposed, some of which are widely used now.
- 9) **Extreme programming (XP)** is one of the most popular methods in agile processes.
- 10) It believes that changes cannot be avoided and rather than treating changes as undesirable, development should accept change.
- 11) To accommodate change, the development process has to be lightweight and quick to respond.
- 12) For this, it develops software iteratively and avoids detailed, multiple documents.
- 13) An extreme programming project starts with user stories.
- 14) **User stories** are short descriptions of scenarios the clients and users experience in the system.
- 15) User stories differ from regular requirements specification & do not contain detailed requirements.
- 16) Requirements are uncovered only when the story is to be implemented, therefore details are decided as late as possible.
- 17) Each story is written on a separate card, so they can be flexibly grouped.
- 18) The empowered development team estimates the time period to implement a user story.
- 19) Using estimates and the stories, release planning is done.



Overall process in XP.

- **Release planning** defines which stories are to be built in which system release, along with their release dates.

- Frequent and small releases are encouraged, hence iterations are used.
- *Acceptance tests* are done for the software before the release.



An iteration in XP.

- Bugs found during the acceptance testing become work items for the next iteration.
- Development is done in iterations, each iteration lasting no more than a few weeks.
- An iteration starts with iteration planning.
- *High-value and high-risk* stories are considered as higher priority and implemented in early iterations.
- Failed acceptance tests in previous iteration also have to be handled.
- Development is done by pairs of programmers instead of individual programmers.
- Automated unit tests be written before the actual code is written, and then the code should be written to pass the tests referred to as test-driven development.
- As functionality of the unit increases, the unit tests are enhanced first, and then the code is enhanced to pass the new set of unit tests.
- It encourages simple solutions and changes.
- *Refactoring* is done to improve the design, and refactored code is used for further development.
- It encourages frequent integration of different units.

- There are many other rules in XP relating to issues like rights of programmers and customers, such as team management, solutions to resolve technical issues, handling bugs etc.

1.6.7 Using Process Models in a Project:

- There are many reasons for using different models.
- While developing software, the purpose is not only to develop software to satisfy the needs of some users or clients, the project should be of low cost and cycle time & should deliver high-quality software.
- In addition, there could be other constraints in a project that the project may need to satisfy.
- Using the constraints of the project, we should use the process model that maximizes the chances of delivering the software, and achieve the highest Q&P.
- Hence, selecting a suitable development process model for a project is a key decision that a project manager has to take.

Let us illustrate this by a few examples.

Example1: A small team of developers is given a task of building a small auction site for a university. The university administration is willing to spend some time at the start to help develop the requirements, but their availability will be limited later. Deadline given is 4 months & there will be no extension of deadline.

- It is clear that a waterfall model is not suitable for this project due to the inflexible deadline.
- The iterative enhancement model is also not right as it requires requirements analysis for each iteration, and the users and clients are not available later.
- The iterative delivery approach where the complete requirements are done in the first iteration but delivery is done in iterations seems well suited, with delivery being done in 2-3 iterations (short time).
- From the requirements, the project team can decide what functionality is essential to have in a working system and include it in the first iteration and other features can be planned for the second iteration.

Example 2: Customers are in a highly competitive environment where requirements depend on what the competition is doing, and delivering functionality regularly is highly desirable. Furthermore, to reduce cost, the customer wants to outsource as much project work as possible to another team in another country.

- For this project, waterfall is not suitable as requirements are not even known at the start.
- Iterative enhancement also may not work as it may not be able to deliver rapidly.
- XP will be hard to apply as it requires that the entire team, including the customer, be involved.
- For this project, the time-boxing model seems to fit the best.
- The whole project can employ three teams—one of analysts will work with the customer to determine the requirements, one to do the development and the third to do the deployment.

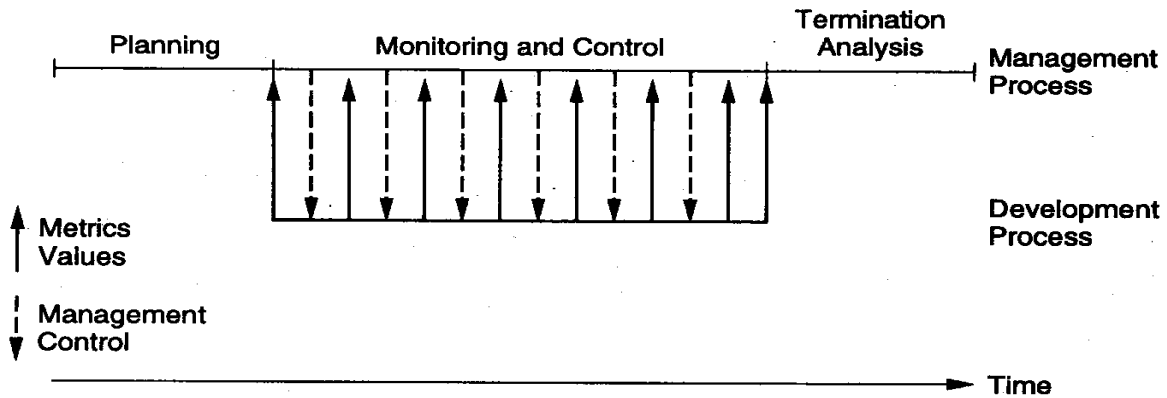
Example 3: A university wants to automate the registration process. It already has a database of courses and pre-requisites, and a database of student records.

- In this project, as the requirements are well understood, the waterfall model seems to be the optimum.

Project Management Process:

- A project management process is necessary to ensure that the engineering process ends up meeting the real-world objectives of cost, schedule, and quality.
- The project management process specifies all activities that need to be done by the project management to ensure that cost and quality objectives are met.
- Its basic task is to plan the detailed implementation of the process for the particular project and then ensure that the plan is properly executed.
- For a large project, a proper management process is essential for success.
- The activities in the management process for a project can be grouped broadly into three phases: *planning*, *monitoring and control*, and *termination analysis*.
- Project management begins with *planning*; the goal of this phase is to develop a plan for software development so that the objectives can be met successfully and efficiently.
- During planning, the major activities are cost estimation, schedule and milestone determination, project staffing, quality control plans, and controlling and monitoring plans.
- Project planning is undoubtedly the single most important management activity.
- *Project monitoring and control phase* of the management process is the longest in terms of duration; it encompasses most of the development process.

- It includes all activities the project management has to perform while the development is going on to ensure that project objectives are met.
- As cost, schedule, and quality are the major forces, most of the activity of this phase revolves around monitoring factors that affect these.



Temporal relationship between development and management process.

- Monitoring a development process requires proper information about the project.
- Such information is typically obtained by the management process from the development process.
- However, interpretation of the information is part of monitoring and control.
- **Termination analysis** is performed when the development process is over.
- The basic reason for performing termination analysis is to provide information about the development process and learn from the project in order to improve the process.
- This phase is also often called **postmortem analysis**. In iterative development, this analysis can be done after each iteration to provide feedback to improve the execution of further iterations.
- **Termination analysis** is performed when the development process is over.
- The basic reason for performing termination analysis is to provide information about the development process and learn from the project in order to improve the process.
- This phase is also often called **postmortem analysis**. In iterative development, this analysis can be done after each iteration to provide feedback to improve the execution of further iterations.