Software:
Software is a collection of programs, procedures, rules, and associated documentation and data
Scenario 1:
- Q: If you have to write a 10,000 line program in C to solve a problem, how long will it take?

Student Says: 2 Months
With 2 months completion time, the prductivity becomes 5000 LOC/person-month
Scenraio 2:
If we pose the same problem to company,
Productivity: 1000 LOC/person-month

Software organization will take 10 person-months to build this software system.
Two different things are being built in two scenarios:

Sr. no	Student System	Industrial Strengh Software System
1	It is built for demonstration purpose	It is built to solve some problem of a client and is used by the client's organization for operating some part of business.
2	Presenace of bugs is nota major issue.	Small bug may lead to huge impactin terms of financial or business loss, inconvience to the users, or loss of property and life.
3	quality is not major issue	System must be of high quality with respect to reliability, usability, portability etc.

1.1 Cost, Schedule and quality:
Software should be produced at **reasonable cost**, in a **resonable time**, and should be of **good quality.**
A. Cost:
Software Size: LOC or KLOC lines of code.
Cost: man power involved
Compay charge to client: $3000 to$15000 per person-month
Productivity is 1000 LOC/person-month
What is cost of 1 LOC= $3 or $15

Small project: 50,000 LOC
Cost: $150000 $ an $750000

B. Schedule: Software needs to be developed faster and within the specified time.

Reducing cost and time are the central goal of software engineering.
Productivity is inversely proportional to cost and schedule.
C. Quality:

Quality consists of 6 major attributes

Software quality:

1. Reliablity: The capability to provide **failure-free service.**
2. Usability: The capability to be **understood, learned,and used.**
3. Portability: the capbility to be **adapted for different specified environment.**
4. Maintainability: the capability to be **modified for purposes of making corrections, improvements or adaptation.**
5. Effficiency: the capability to provide **apropriate performance** relative to the amount of **resources used**.
6. Functionality: the capability to provide functions which meet **stated and implied needs when software** is used.

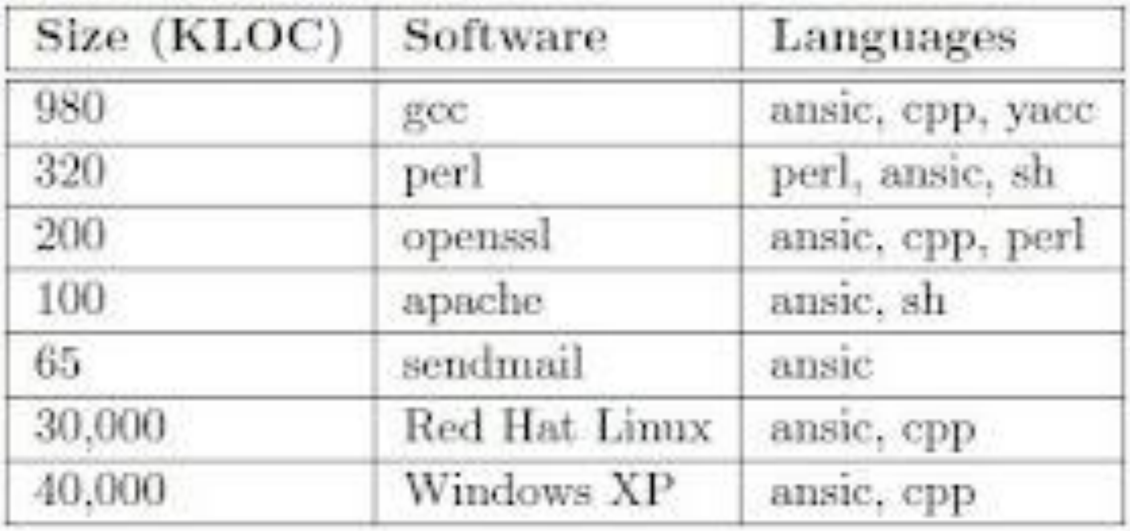
C.Quality of software:
To find the number of defects in the software that was delivered (log defects after 6 months)

Mantainance:

- Software always has residual errors.
- These defects once discovered, need to be removed, called **corrective maintainance.**
- Softwar is further satifying enhaced needs of the users and the environment called **adaptive maintainance.**

1.2 Scale and Change

- There are two characteristics of problem domain that influence the solution approaches used. Scale and change
- Industrial strength software systems are generally large and complex. It requires tens of thousands of LOC.
- Sizes of the well known software products are given below.



Size (KLOC)	Software	Languages
980	gcc	ansic, cpp, yacc
320	perl	perl, ansic, sh
200	openssl	ansic, cpp, perl
100	apache	ansic, sh
65	sendmail	ansic
30,000	Red Hat Linux	ansic, cpp
40,000	Windows XP	ansic, cpp

Counting problem:

- Problem of counting people in a room vs taking census of a country
- Different methods wil have to be used for conducting census. It requires more management, organization and validation.
- Method which is used for writing code of 100 LOC will not work for writing code of KLOC.
- In small project, informal methods for development and management can be used. However, for large project both have to be much rigorous.
- For successful project, proper development methods and tightly management is required to make sure that cost, quality and schedule are under control.

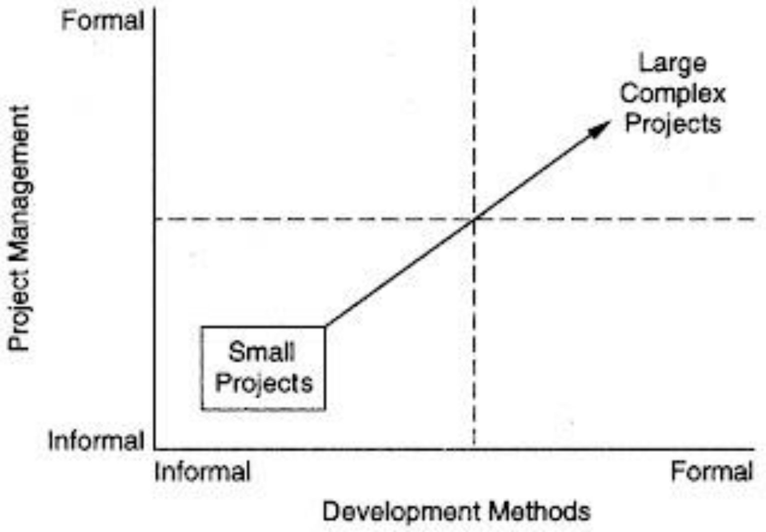


Fig. problem of scale

Change:

- Change is another characteristic of the problem domain which the approaches for development must handle.
- At intial stage requirements are not known, as development proceeds, and the time passes, additional requirements are identified.
- These changes must be incorporated in the software. If change request is not handled properly, can consume up to 30-40% of the development cost.

1.3 Software Processes

- **Software Engineering**: is a **systematic approach** to the **development, operation, maintainance, and retirement** of software.

- Goal of sofware engineering is to achieve **high quality, low cost and low cycle time**. Systematic approach must help to achieve a **high quality and productivity (Q&P)**.

- In software, three main factors that influence Q&P are **people, processes, and technology**.

- Finaly **quality delivered and productivity [Lines of code per staff per month] achieved** depends on the **skills of the people** involved in the software project, the **processes people use to perform** the different tasks in the project, and the **tools** they use.

- Software engineering focuses on **the process** for producing product.

- Process: is a **sequence of steps** performed for a **given purpose**.
- Purpose is to develop software to satisfy the needs of some users or client.
- Software process is a **sequence of steps** performed to develop software which satisfy the needs of some users or client.

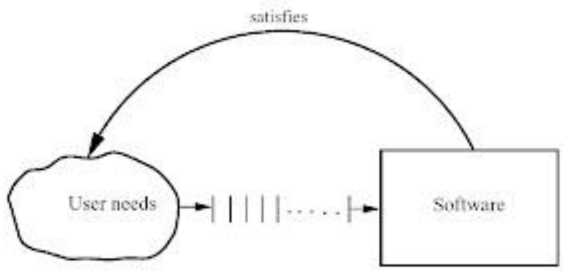


Fig. Basic Problem

- To achieve **High quality and productivity** we need optimum process.
- **A process model** specifies a general process, which is “**optimum”** for a class of projects.
- Process means to **reach the goals of high quality, low cost and low cycle time**, and a process model provides a **process structure** that is well suited for a class of projects.

Software Processes

- The **processes** that deal with the **technical and management issues** of the software development are collectively called the software process.
- Component of software process
    A. **Development process**: Specifies all engineering activities that need to be peroformed.
    B. **Project Management Process**: Specifies how to plan and control these activities so that cost, schedule, quality and other objectives are met.
- During the project many products are produced which typcially composed of many items (Final source code may be composed of many source file).
- Many **versions are generated** as project proceeds, these evolutions and changes are handled by another process called **software configuration control process.[Role is to manage changes]**
- Role of **process management process is** to **improve software process.**
    - Capability of process to produce good quality goods at low cost is improved.
    - It includes understanding of current process, analyzing its properties, determining how to improve, and then affecting the improvement.

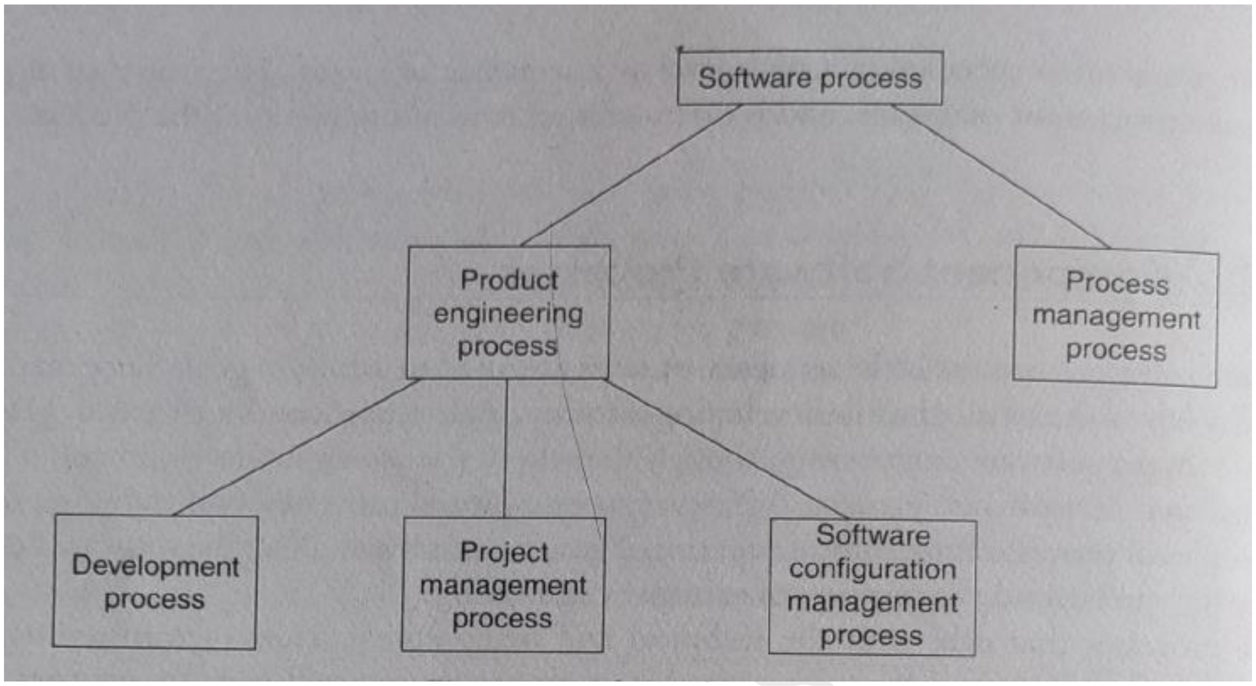

Software process
Product engineering process
Process management process
Development process
Project management process
Software configuration management process

Fig. software process

SR.NO	Process activities	Person/Group
1	Development Process activities	programmar, Designer and tester
2	Project management process activities	Project management
3	Configuration control process activities	Configuration Controller
4	Process managment process activities	Software Engineering Process Group(SEPG)

1.4 Software Development Process Models
1. Waterfall Model
2. Prototyping
3. Interative Development
4. Timeboxing Model

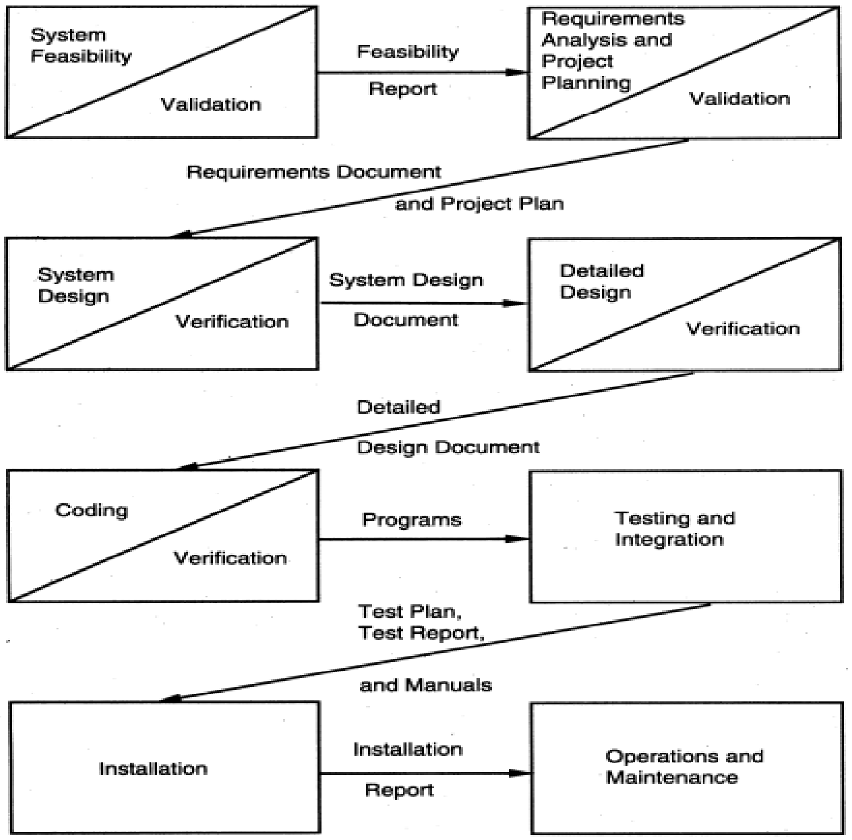
1. Waterfall Model

- Developed by Royce
- In this model project begins **with System feasibility analysis**. Upon successfully demonstrating the feasibility of a project, Requirement analysis and planning begins. The **coding begins after design is complete. Once programming is completed, code is integrated and testing is done. Once testing is done, system is installed. After this regular maintainace of the system takes place.**
- **Requirement analysis and planning**- planning is a critical activity in a software development. A good plan is based on the system requirement.

- **Work product**: output of every phase is called work product.

Each phase must have defined output that can be evaluated and rectified

- Requirements document
- Project plan
- Design documents (architecture, system, detailed) –
- Test plan and test reports
- Final code
- Software manuals



The Waterfall Model

Sr. NO	**Phase**	**Work product**
1	Requirement Analysis	Requirements document
2	Project planning	Project plan
3	System Design	System Design document
4	Detailed Design	Detailed Design document

5	Coding	Program
6	Testing and integration	Test plan and test reports and manuals
7	Installation	Installation report
8	Operation and Maintainance	

Advantages:

- The **simplest process Model**: **Straightforward.** Divides large task of a building software system into series of cleanly devided phases. Phases are organized in a linear order.

- **Easily manageble**: because once **particular phase is get finished** developing orgainzation can demand money from customer.

Limitation:

1. **It assumes that the requirement of system can be frozen before design begins**. [This is not possible for new system]
2. **Freezing the requirements usually requires choosing the hardware.** [large project might take a few years to complete. If the hardware is selected early then due to the speed at which harware technology is changing, final software will use obsolete hardware technology.]
3. It follows **the "big bang" approach**. [entire software is deliverd in one shot at the end.[This includes heavy risk, because user don't know much about requirement]
4. It encourages "**requirement bloating**". [user might add such requirements which not may not get used]
5. It is **document driven process** [Requires formal documentation after every phase.

Prototyping model:

- The goal of protoyping based model is to **counter the first limitation of the waterfall model.**
- Instead of freezing the requirement before any design or coding can proceed, a **throwaway prototype** is built to help **understand the requirement**.
- One prototype is built based on **current requirement.**
- In Prototype **all phases are carried** out but not in **formal way**
- By prototype model **client gets actual feel of system** so that client can understand the **requirement of desired system**. This leads to the better **stable requirements**.

Prototyping is an **attractive idea** for **complicated and large systems** for which there is no manual process or existing system to help to determine the requirement.

Based on the preliminary requirement and specification document prototype is developed. End users and clients are given an opportunity to use and explore a prototype. Based on their experience, they provide a feedbackto the devlopers regarding prototype.

1. What is correct
2. What is need to be modified
3. What is missing
4. What is not needed

Based on the feedback **prototype is modified** to incoroporate changes that can be done easily.then users are gained access to use the system. This is done iteratively till final requirements are understood.

Advantages:

1. As requiremets are uderstood properly, it reduces overall development cost.
2. Requirements will be **more stable** due to feedback from the prototype. There will be **fewer changes in ther requirement**. Cost of requirment chage will be reduced.
3. **Quality software** will be produced. As requirements are understood properly, better design, codig and testing will be done.
4. It **reduces the risk** associated with project.

<u>3. Iterative Development</u>:

Counters third and forth limitation of waterfall model

1. It follows **the "big bang" approach**. [entire software is deliverd in one shot at the end.[This includes heavy risk, because user don't know much about requirement]
2. It encourages "**requirement bloating**". [User might add such requirements which may not get used].

Combines advantages of both waterfall and protyping model
Basic idea:
Software should be developed **in increment**, after each increment **new additional capability** to be added in software till final implementation.

First step: simple implemetnation is done **for subset of the overall problem**.

Project control list: is prepared that contains, in order, **all the taskes** that must be performed that must be performed to obtain the final implementation.

Every iteration, one task is removed from the list for which design, implemantion and analysis phase is carried out. The process is iterated until the project control list is empty.

In final iteration implemenation of system will be available.

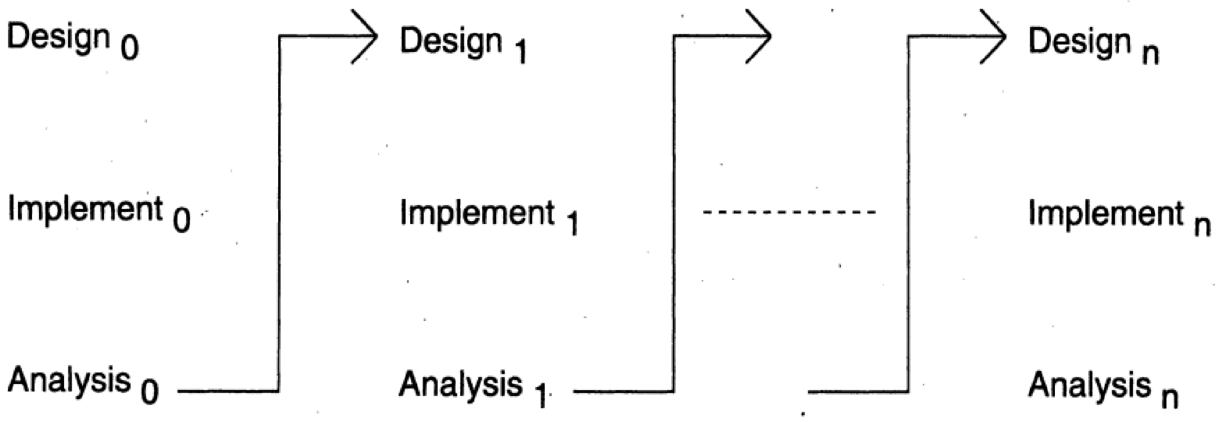


Figure 2.5: The iterative enhancement model.

Project control list guides the **iteration steps** and **keeps track of all tasks** that must be done. Task is selected in such a way that it minimizes the chances of errors and reduces the redesign of work.

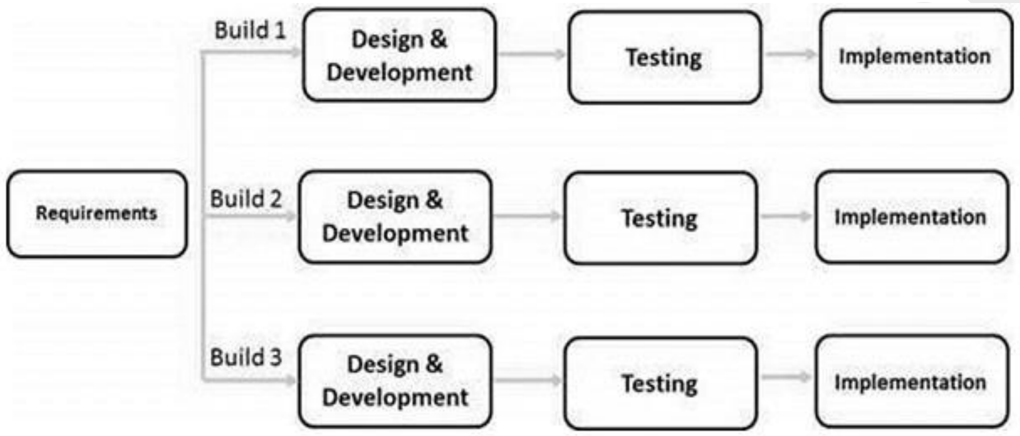
Another approach for **iterative development** is to do the requirements and architecture design in a standard waterfall or prototyping approach, but deliver a software iteratively. [in the first iteration,which requirement is to be satisfied,is decided and design impemenation is done]

Advantages:

1. Requirements are known and overall view of system is available and a proper stable architecture can be designed.
2. It doesn’t have the all-or-nothing approach.

Approach is more popular:

1. Clients do not want to invest too much without seeing reutures. In iterative model, after each iteration, working software is delivered
2. Businesses are changing rapidaly today; they never really know the complete requirements of software. Adding new functionality based current situation of business is important.
3. Each iteration provides a working system for feedback, which helps in developing stable requirements for the next iteration.



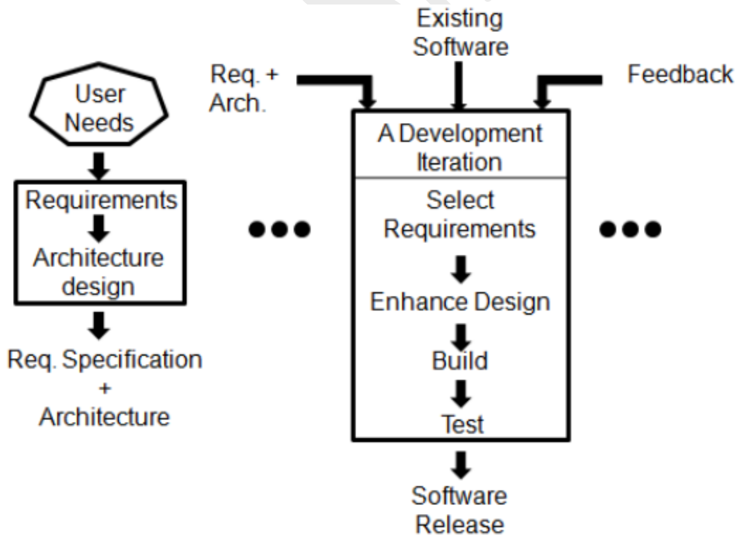


Figure 2.6: Iterative delivery approach.

<u>Rational Unified Process (RUP):</u>

It is another iterative process model that was designed by Rational now part of IBM.
Designed for Object oriented development using UML

Development of software is devided into **cycles,** each cycle delivering a fully **working system.** Each cycle is executed as **separate project.**

Each cycle it self is broken into the following four Consecutive phases:

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

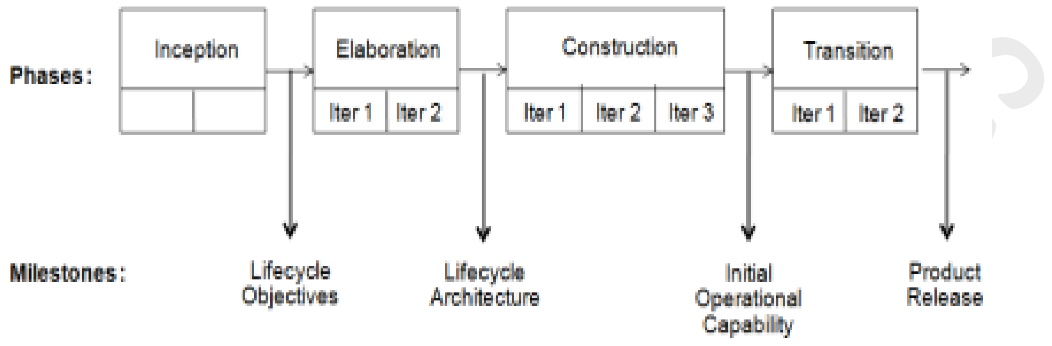


Figure 2.7: The RUP model.

Each phase has distinct:

1. Purpose
2. Completion of phase [milestone]

Inception phase:

Purpose: to establish goals and scope of the project [vision,scope of project, key risks,basic plan regarding project cost and schdule]

Completion of phase: lifecycle objectives

Elaboration phase: purpose: to design architecture of a system

Completion of phase: lifecycle architecture [most of the requirements have been identified and specified]

Construction phase: software is build and tested.

Completion of phase: initial operational capability

Transition phase: purpose: to move the software from the development environment to the clients environment where it is hosted.

Completion of phase: product release

Each phase itself may have muliple iterations. Construction phase will be broken into multiple iterations, each iteration producing a working system which can be used for feedback, evaluatio, beta-resting etc.

RUP group the activities into different subprocesses which it calls core process **workflows**.

These Subprocesses correspond to the tasks of performing requirements analysis, doing design, implementing the design, testing, prject management etc. subprocess are shown in table.I

4. Time boxing model:

Basic idea:

Software should be developed **in increment**, after each increment **new additional capability** to be added in software till final implementation.

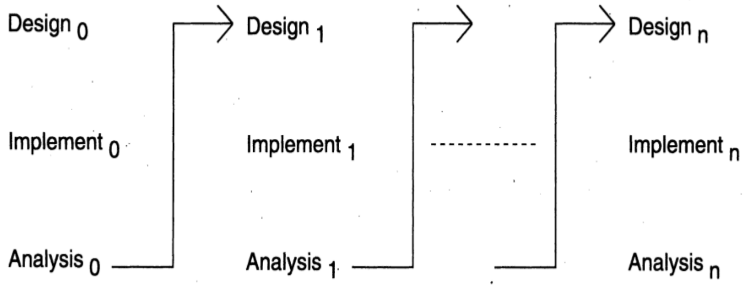


Figure 2.5: The iterative enhancement model.

Limitation of Iterative Development Model:

1) Single team work on the all phases of the project
2) Time consuming. It takes **long development time** to complete project.

How to speed up development time?

Time boxing model:

Parallelism concept is used. Iterations are executed in parallel fashion instead of sequential manner. **New iteration** starts **before the system** produced by the **current iterationis released**.

New iteration starts before the current iteration ends. By doing this overall, it is possible to reduce overall average delivery time for iterations.

- Basic unit of development is **time box,** which is of **fixed duration**.
- Each time box is devided into a **sequence of stages** [waterfall model].
- Each stage perform clearly **defined task for iteration** and produces **defined output.**
- During fixed time the task is completed
- Tasks in the stages are performed by **dedicated team** [Anlyst team, build team, deplyoment team]

Now consider the time boxing model consisting of three stages:

Stages	**Task**	**Team**
Analyst and Requirement	Prioritized list of requirements and high level design	**Analyst**

specification		
Build	Develop code and Do testing	**Developer, Tester**
Delplovment	Predefined deployment tests and installtion	**Technical suport**

All stages are done in **equal time.**

With time box of three stages, project proceeds as follows

- When the requirements **team has finished requirement for timebox-1**, the requirements are given to **build team** for building the software. The requirement team then goes on and starts **preparing the requirements** for **timebox-2(TB2)**
- When the **build for TB1 is completed**, the code is handed over to the deployment team, and build team **moves on** to build code for requirement for TB2.
- Requirement team moves on to **prepare requirement for TB3.**

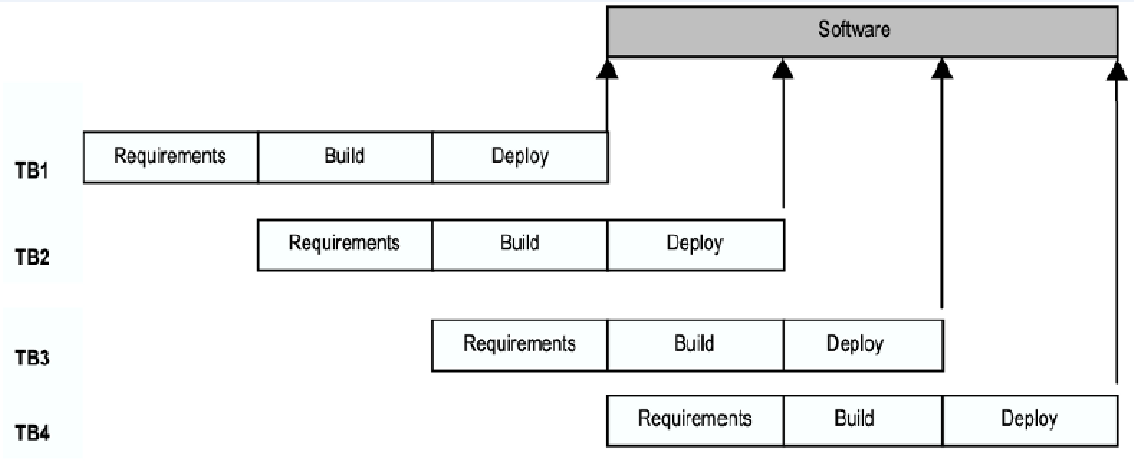


Fig. executing the timeboxing process model.

With three **stage time box**, three iterations can be concurrently in progress. TB duration T is **9 Weeks** (**Each stage duration is 3 weeks**), the first delivery is made **9 weeks** after the start of the project.
Second delivery after **12 weeks, third** delivery after **15 weeks**.

Linear execution of iteration: first delivery is made after 9 weeks, second after 18 weeks, third after 27 weeks.

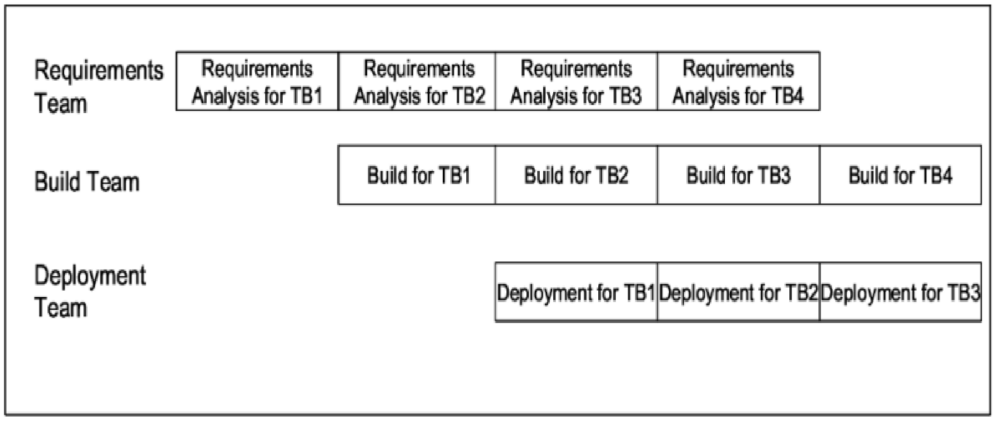


Figure 2.9: Tasks of different teams.

Advantages:

- **Delivery time is reduced** by using additional manpower.
- This model is suited for the projects that requiresa large number of features to be developed in a short time.

Disadvantages:

- Increases **complexity of project management**. [Additional cost]

Extreme Porgramming and Agile Processes:

- Evolved in 1990s

Principles of Agile development aproach

1. Our highest priority is to **satisfy the customer** through early and **continuous delivery of valuable** software.
2. Welcome **changing requirements**, **even late in development**. Agile processes harness change for the customer's competitive advantage
3. **Deliver working software** frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. **Business people and developers** must work together daily throughout the project.
5. **Build projects around motivated individuals**. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation**.
7. **Working software** is the primary measure of progress. [This mentality pushes to get products to the market quickly rather than let documentation]

8. Agile processes **promote sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to **technical excellence and good design enhances** agility
10. **Simplicity**—the art of maximizing the amount of work not done—is essential [doing the things that can have the most impact]

Many detailed methodologies have been proposed,
Extreme programming (XP) popular approach in the family of agile methods.

1. User stories

- XP Project starts with **user stories** [short user or customers need]
- User stories **do not** contain **detailed requirements**
- **Acceptance test** is also built from stories.
- Development team **estimates time** to implement user story. Estimates are rough [weeks]

2. Release planning: decided **which stories are** built in which system release [dates of releases]
3. Iteration: Development is done in interation. It starts with **iteration planning.**
   Stories are implmented in each iteration.

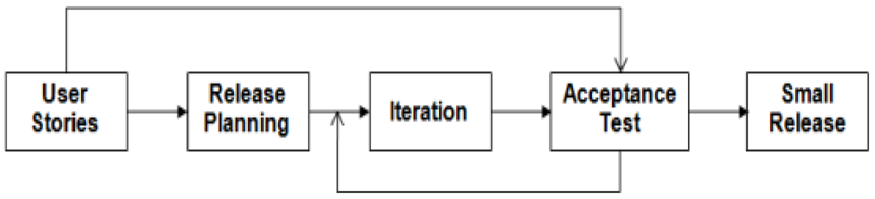


Figure 2.10: Overall process in XP.

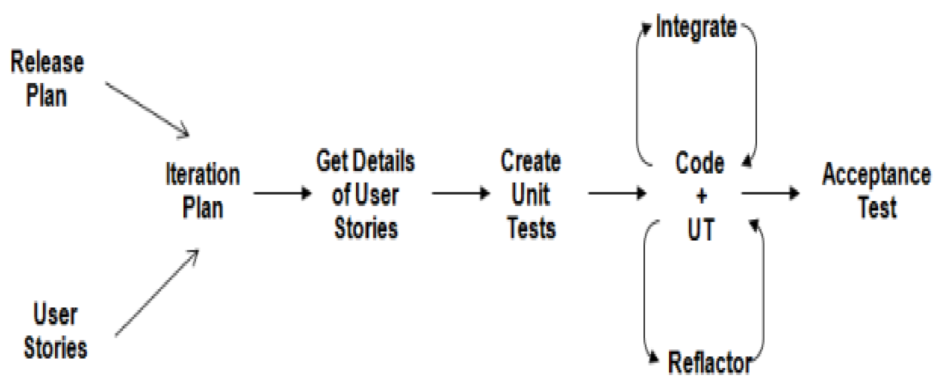
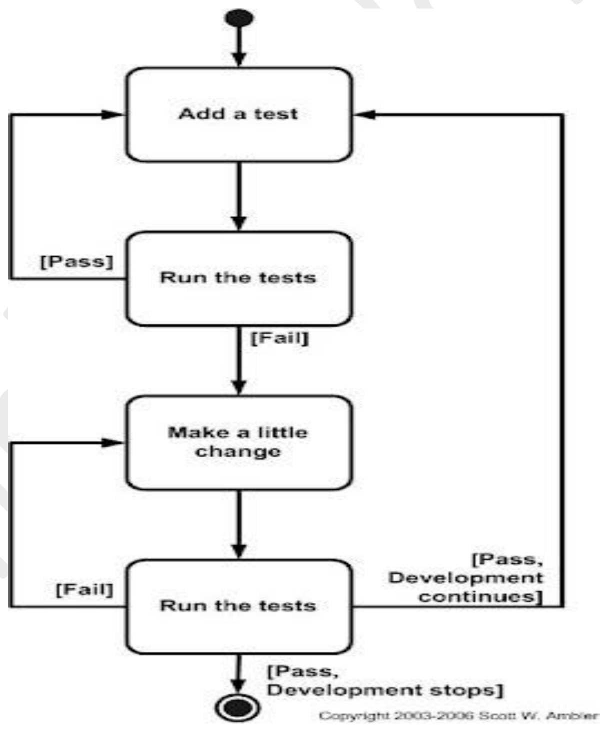


Figure 2.11: An iteration in XP.

Development approach used in iteration is called **test driven aproach.**
Unit tests are enhanced first **and** then the **code is enchanced** to pass **new set of unit tests.**


Add a test
[Pass]
Run the tests
[Fail]
Make a little change
[Fail]
Run the tests
[Pass, Development continues]
[Pass, Development stops]
Copyright 2003-2006 Scott W. Ambler

In TDD, developers start **creating small test cases** for **every feature** based on **their initial understanding.** The primary intention of this technique is to **modify or write new code** only if the tests fail. This prevents duplication of test scripts.

1. **Create precise tests**: Developers need to **create precise unit tests** to verify the functionality of specific features.

2. **Correcting the Code**: Once **a test fails**, developers need to make the **minimal changes** required to correct the code so that it can run successfully when re-executed.
3. **Refractor the Code**: Once the test runs successfully, check for redundancy or any possible code optimizations to enhance overall performance. Ensure that refactoring does not affect the external behavior of the program.

csUnit and NUnit – Both are open source unit testing frameworks for .NET projects.
PyUnit and DocTest: Popular Unit testing framework for Python.
Junit: Widely used unit testing tool for Java
TestNG: Another popular Java testing framework. This framework overcomes the limitations of Junit.
Rspec: A testing framework for Ruby projects

1.5 Project Management Process:

Development process: decides **phases and tasks to be done**

Doesn't specify **duration** of phase, how many **resources** should be **assigned** to each phase, how a phase should be **monitored.**

To meet cost, quality and schedule objectives, r**esources** have to be **properly allocated** to each activity for the project, and **progress of different activities** should be **monitored** and if **objectives are not met** corrective **actions** should be taken.

Three activities are performed in management process:

1. Planning:
    - Goal is **to plan** for software development.
    - It is produced **before development begins**; plan is get updated as development proceeds and data about progress of project is available.
    - **Cost estimation, schedule and milestone determination, project staffing, quality control plans, and controlling and monitoring plans.**
2. Monitoring and control:
    - **Longest term** of duration
    - It checks whether **project objectives are met** as development proceeds.
    - Development proceeds according to **developed plan.**
    - **Cost, schedule and quality** factors are to be monitored during development.
    - Monitor the **risk** for the project.

- Information obtained in **monitoring phase** suggests that objectives may not be met; **necessary actions** are taken by imposing **suitable control** on the development activities.
- Development process provides the information to management process it needs.

3. Termination Analysis/Postmortem analysis:
   - Peformed when developmet process is over
   - Goal: is to provide information about the **development process** and **learn from the projec**t in order to **improve the process.**

Fig shows that, during development, **quantitative information** flows to the monitoring and control phase of the management process, which uses i**nformation** to **exert control** on the development process.

Software development metrics are **quantitative measurements of a software product or project**, which can help management understand software **performance, quality, or the productivity and efficiency** of software teams.

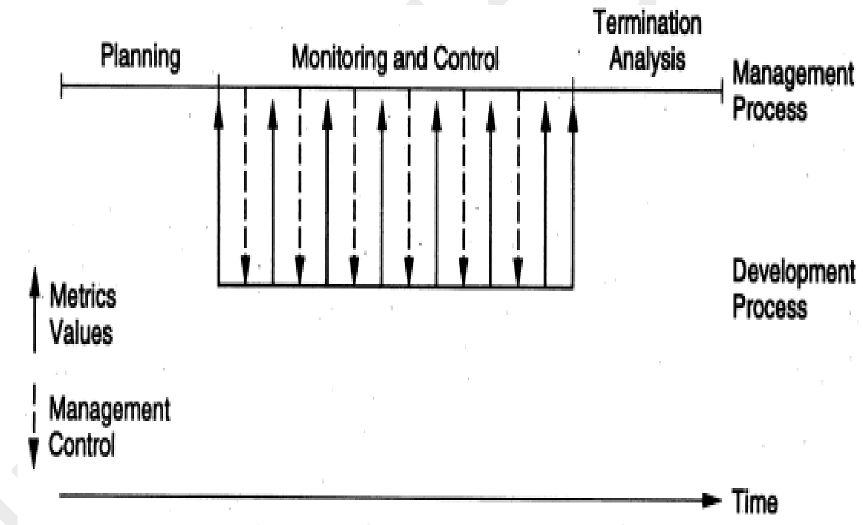


Fig.Temporal relationship between development and management process