UNIT 1

## * The Software Problems *

Date:

Introduction :-

A student system is being built which is primarily meants for demonstration purposes, and is not expected to be used later. Because it is not to be used, nothing of significance depends on the softwar & the presence of bugs and lack of quality is not major concern. Neither are the other quality issues like usability, maintainability, portability etc.

On the other hand, an industrial-strength software system is built to solve some problem of a client & is used by the client's organization for operating some part of business, and a malfunction of such a system can have huge impact in terms of financial or business loss, inconvenience to users, or loss of property & life. Consequently, the software system needs to be of high quality with respect to properties like reliability, usability, portability etc.

- The software industry is largly interested in developing industrial-strength software, and the area of software engineering focuses on how to build such systems. That is, the problem domain for software engineering is industrial-stregth softwar,

1.1 Cost, schedule & Quality :-

In the industrial streyth software domain, there are ~~three~~ basic forces at play- cost, schedule & quality.

The S/w should be produced at reasonable cost, in a rasonable time, and should be of good quality.

Cost :-

Industrial-strength software is very expensive primarily due to the fact that software developement is extremely labor-intensive.

LOC (Lines of code) or thousands of lines of code (KLOC) delivered is by far the most commonly used measure of software size in the industry. As the main cost of producing software is the manpower employed, the cost of developing software is generally measured in terms of person-month of effort spent in development.

The productivity is frequently measurd in the industry in terms of LOC (or KLOC) per person-month.

The productivity in the software industry for writing fresh code generally ranges from few hundred to about 1000 + LOC per person-month.

schedule :-

It is another important factor in many projects.

The cycle time from concepts to delivery should be small.

for software, this means that it needs to be developed faster & within the specified time.
- Unfortunately, the history of software is full of cases when projects have been substantially late.
- Therfore, reducing the cost & 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. (the same work can be done with fewer person-months)

Quality:-
- It is third important factor of software engineering.
- Developing high-quality software is another fundametal goal of software engineering.
- while cost is generally well undestood, the concept of quality in the context of SW needs further elaboration.
- The international standard on software product quality suggests six main attributes:-

① Functionality:-
The capability to provide functions which meets stated & implied needs when the s/w is used.

② Reliability:-
The capability to provide failure-free service.

③ Usability:-
The capability to be understood learned and used.

4. Efficiency :-
The capability to provide appropriate performance relative to the amount of resources used.

5. maintainability :-
The capability to be modified for purposes of making corrections, improvment or adaptation.

6. portability :-
The capability to be adapted for different specified environments without applying actions or means, other than those provided for this purpose in the product.

- With multiple dimensions to quality, different project may emphasize different attributes & a global single number for quality is not possible.
- As unreliability of software is due to the presence of defects in the software, one measure of quality is the number of defects in the delivered software per unit size.
- Once the software is delivered & deployed it enters into the maintenance phase.

- Corrective maintenance
- Adaptive maintenance.

- Over the life of software system, maintenance cost can far exceed the cost of original developement.
- The maintenance-to-development-cost ratio has been variously suggested as 80:20

70:30, or 60:40

* Scale & change :-

i) Scale :-

— As can be expected, development of large system requires a different set of methods compared to developing small system.

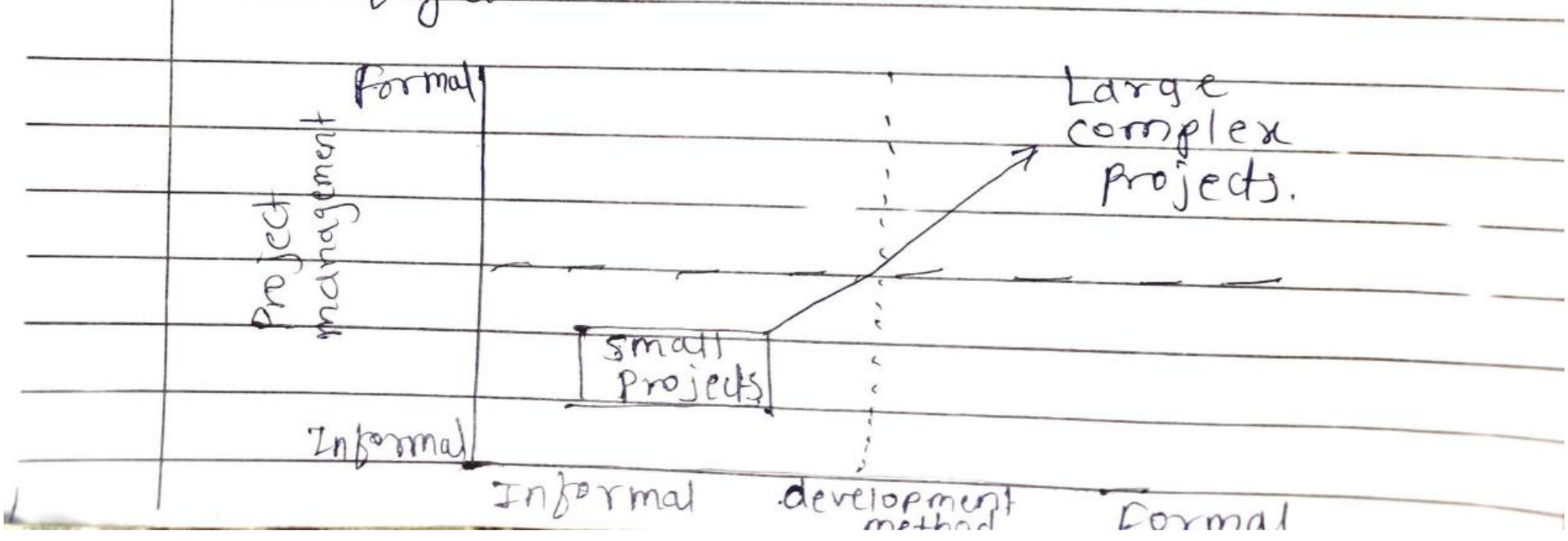
— The methods that are used for developing small systems often do not scale up to large systems.

eg. cosider a problem of counting people in room. Vs taking a censw of a country.

— Any software project involves the use of engineering & project management.

— In small projects, informal methods for development & management can be used.

— For large scale projects, both development & management have to be much more rigorous as shown in figure.



ii) change :-

change is another characteristics of the problem domain
- The complete set of requirements for the system is generally not known or stated.
- As the development proceeds & time passes, additional requirements are identified.
- That needs to be incorporated in the software being developed.
- chang requests can be quite distuptive to a project & if not handled properly, can consume up to 30-40 % of the development cost.
- Software has to be changed even after it has been deployed.
- The world changes faster, s/w has to change faster, even while under development.

## Scale of change:-

- Development of large system requires a different set of methods compared to developing small system.
— As the methods that are used for developing small systems often do not scale up to large systems.
- eg. Consider the problem of counting people in a room. Vs. taking census of country.

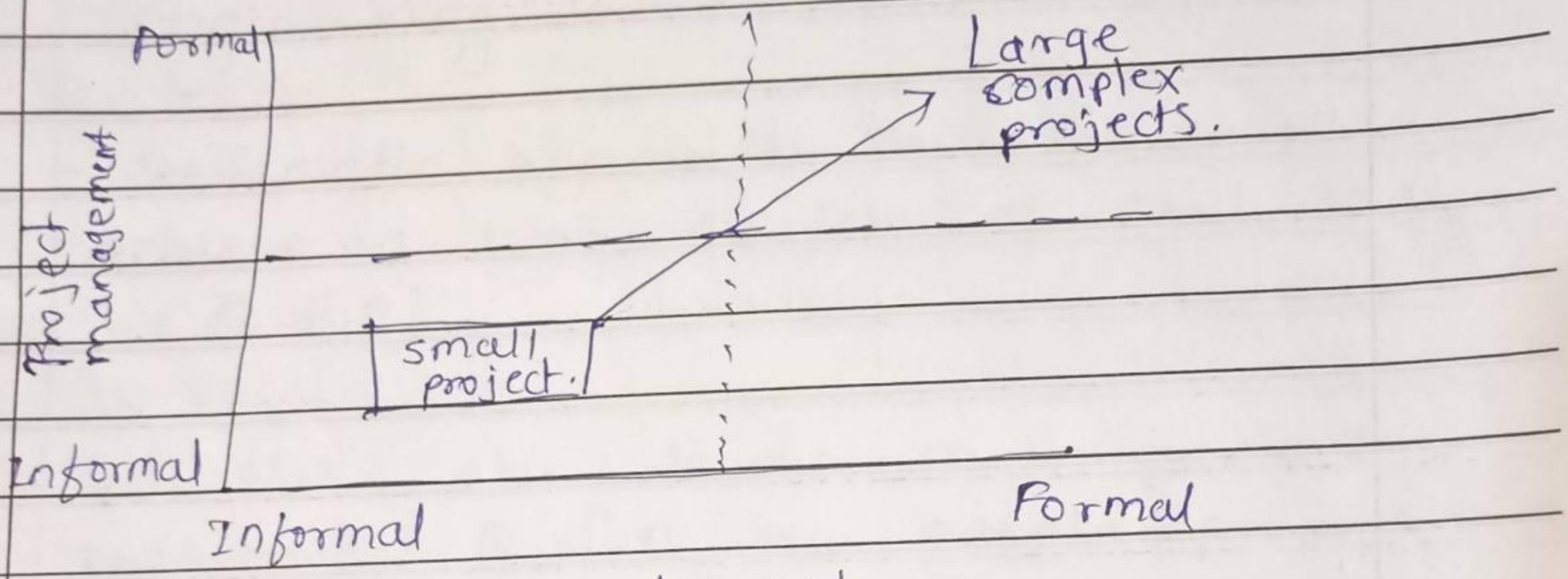



fig. The problem of scale.

## * Software Processes :-

→ Definition :- Software Engineering :-

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

- Beside delivering s/w, high quality, low cost & low cycle time are also goals which s/w engineering must achieve.

- Systematic approach must help to achieve a high quality & productivity (Q & P).

- In s/w, the three main factors that influence Q & P are people, processes & Technology.

- People who ultimately develop & deliver, the main job of processes is to help people achieve a higher Q & P by specifying what task to do & how to do them.

- Tools or technologies are aids that help people perform some of the tasks more efficiently & with fewer errors.

## Process & Projects:-

### i) Process:-

- A process is a sequence of steps performed for a given purpose
- while developing software, the purpose is to develop software to satisfied the needs of some users or clients as shown in fig.

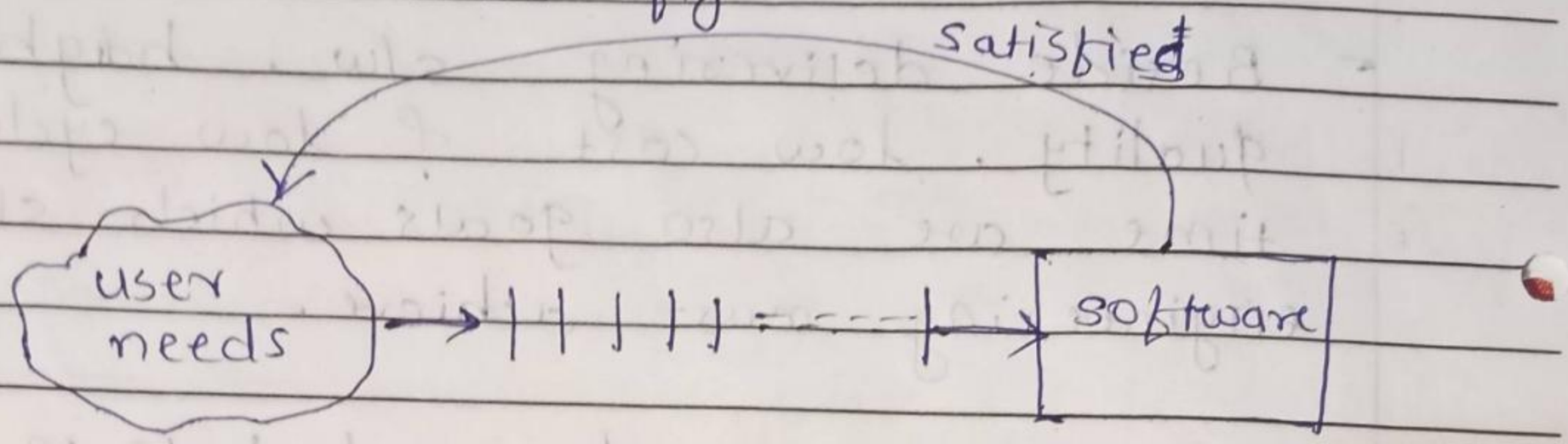



fig. Basic problem.

### ii) Software project:-

- A software project is one instance of basic problem & development process is what is used to achieve this purpose.
- For a project its development process plays key role.
- The role of process increases due to additional goals like, low cost, in low cycle time & it delivers high quality softwares.
- Many processes can achieve the basic goal of developing software

Process model:-

- A process model specifies a general process, which is "optimum" for a class of projects.
- A process model is essentially a compilation of best practices into "recipe" for success in the project.

* Component s/w processes:-

Software process:-

The process that deals with the technical & management issues of software development are collectively called the Software process.

Two major components:-

1. A development process
2. Project management process.

- The development process specifies all the engineering activities that need to be performed.
- Management process specifies how to plan & control these activities so that need cost, schedule, quality & other objectives are met.

- As development processes generally do not focus on evaluation of changes, to handle them another process called software configuration control process is often used.
- The objective of this component process is to primarily deals with managing changes, so that the integrity of the products is not violated despite changes.

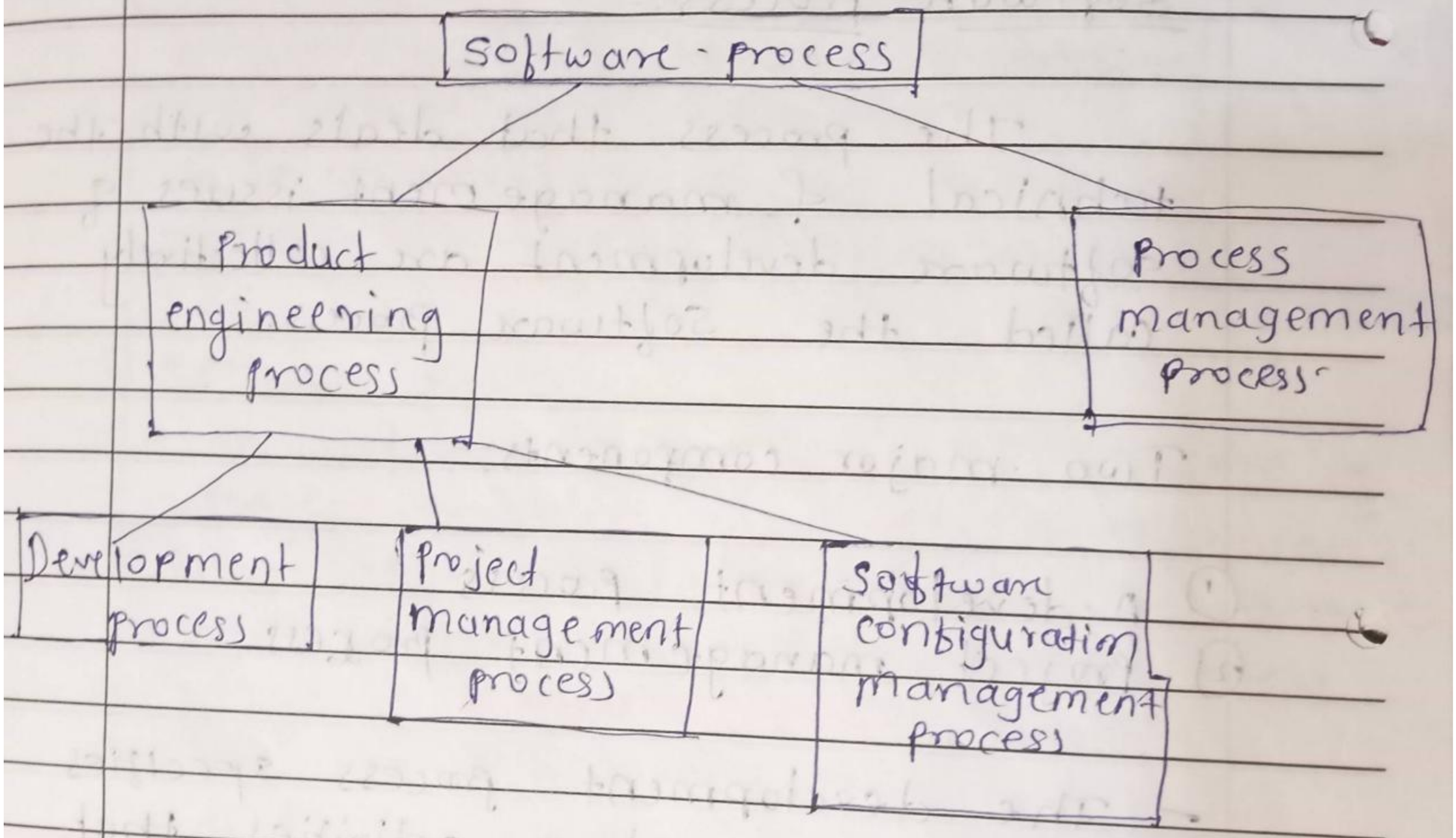



fig: Software processes.

The relationship between these component processes is shown in figure

## * Software development process model:-

- The goal of software development process model is to produce a high-quality software product.
- A process model specifies a general process, usually as a set of stages in which a project should be divided.

There are five software development process model:

### (1) Waterfall Model:-

- The simple process model is the water-fall model.
- The phases are organized in a linear order.
- Model was proposed by Royce.
- In this model, a project begins with feasibility analysis.
- Upon successfully demonstrating the feasibility of a project, the requirement analysis & project planning begins.
- The design starts after the requirements analysis is complete,
- And coding begins after the design is complete.
- Once the programming is completed, the code is integrated and testing is done.
- Upon successful completion of testing the system is installed.
- After this regular operations & maintainance.

of the system takes place.
The model is shown in following figure.

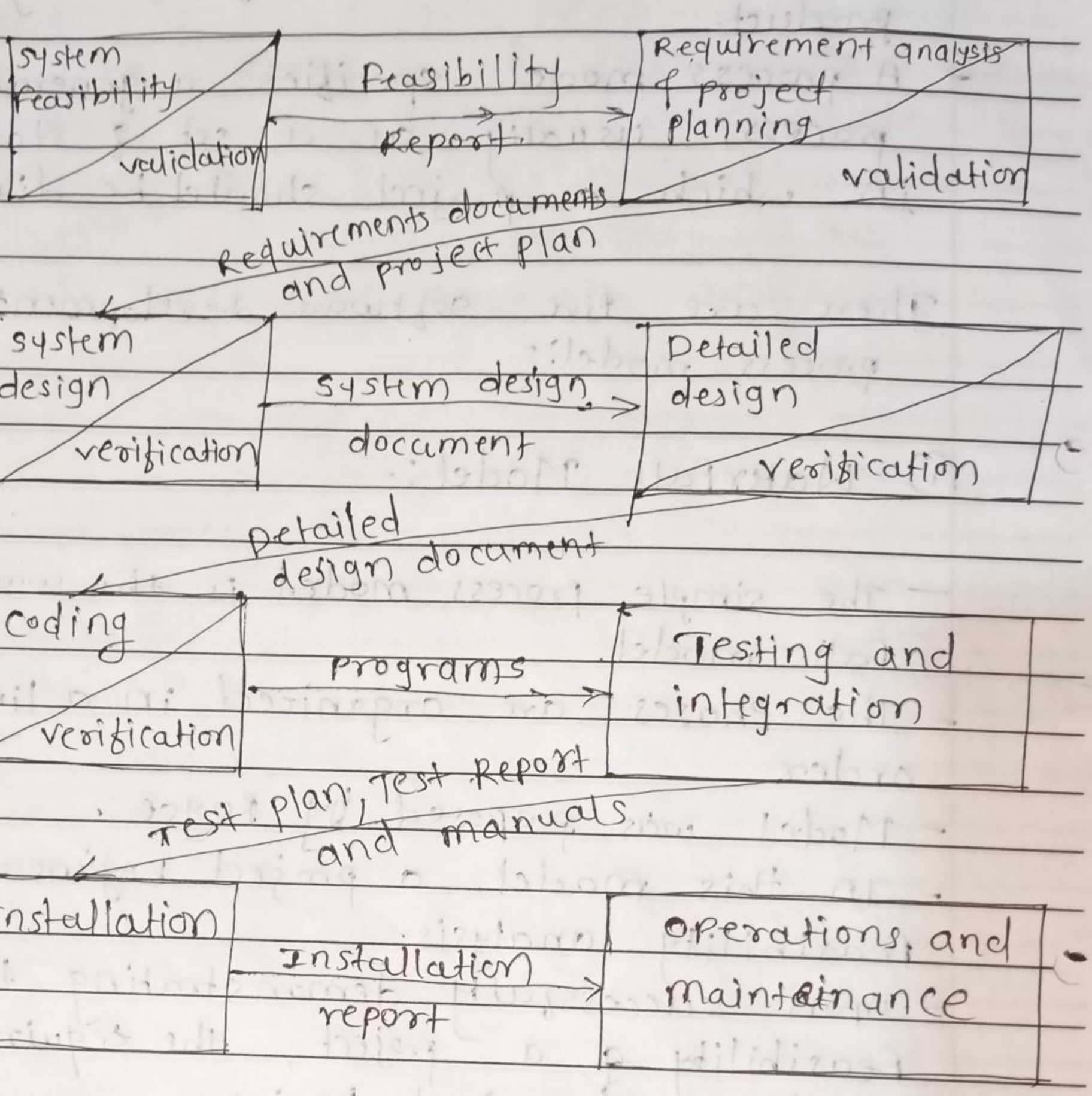



figure :- The water fall model.

- The basic idea behind the phases is seperation of concerns.
- The large & complex task of building the softwar is broken into smaller tasks.
- Linear ordering of activities has some important consequences.

- 1st to crealy identify the end of the each phase
- when the activities of phase are completed, there should be some product that is produced by that phase.
- Following documents should be produceed in each project:

1) Requirement document
2) Project plan
3) Design documents (architecture, system, detailed)
4) Test plan & test reports
5) Final code
6) Software manual (eg user, installation etc)

Advantages :-

1) The main advantage of the waterfall model is its simplicity.
2) It is conceptually staightforward & divides the large task of building a software system into series of cleanly divided phases.
3) easy to administer in a contractual setup

Limitations:-

1) It assumes project requirements are final & not going to change; but in reality project requirements changes day by day.
2) waterfall model requires hardware specificat^n to be freezed in early stage of project. which is problematic when project complets after few years as that time the faster hardware will be available.

3) It follows big bang approach — entire software is delivered in one shot. This is risky, as user doesn't know what he is going to get at the end.
4) It encourages "requirements bloating". All the requirements must be specified at the start & only what is specified will be delivered.
5) It is a document-driven process that requires formal documents at the end of each phase.

## 2) Prototyping model:-

- The goal of prototyping-based development process is to overcome the limitation of the waterfall model.
- The basic idea is that, instead of freezing the requirements before any design or coding can proceed.
- A throwaway prototype is built to help understand the requirements.
- This prototype is developed base on the currently known requirements.
- Using this prototype, the client can get an actual feel of the system, which can enable a client to better understand the requirments of desired system.
- Prototyping is an attractive idea for complicated & large systems.
- It is also an effective method of demonstrating the feasibility of certain approaches.

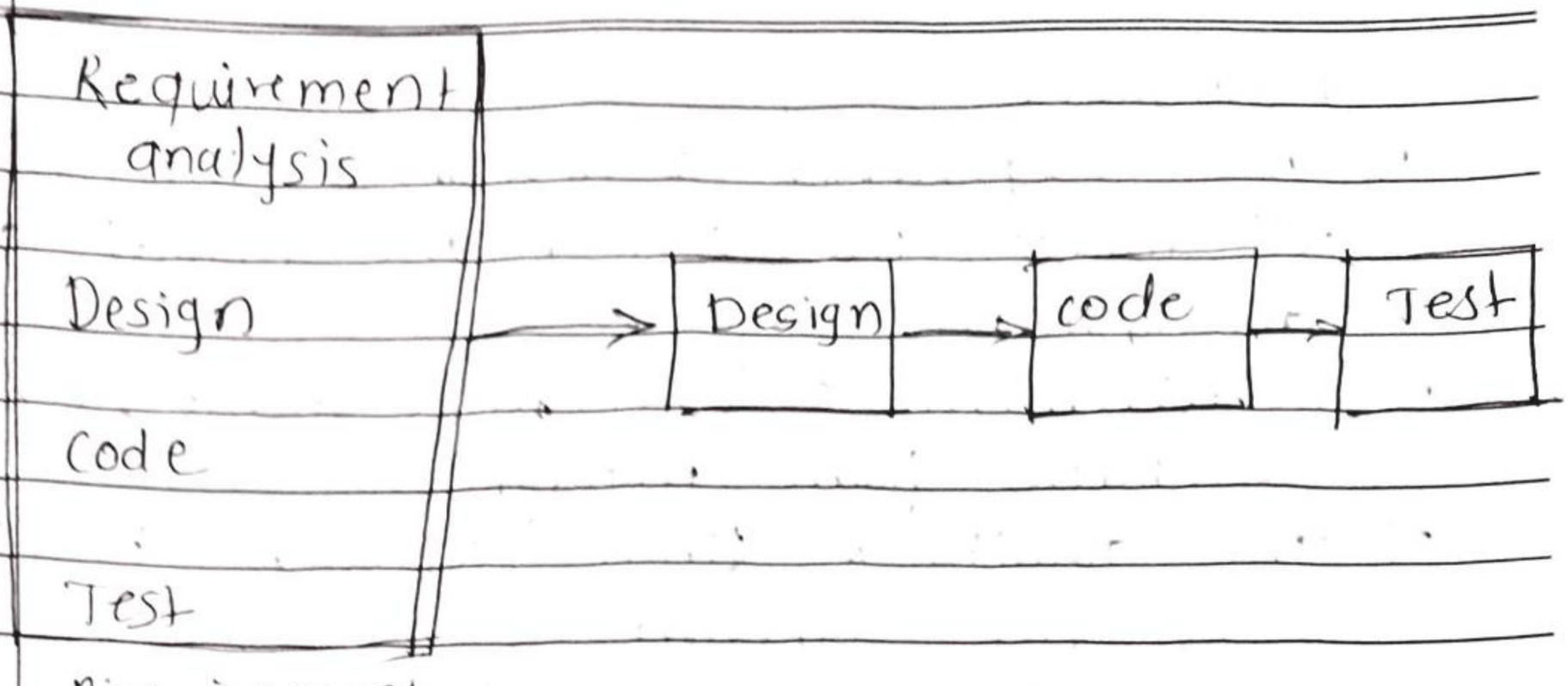



fig. The prototyping model.

- The development of the prototype typically starts when preliminary version of the requirements specification document has been developed.
- After prototype has been developed, the end users, clients are given an opportunity to use & explore prototype.
- Based on their experience, they provide feedback to the developers.
- Then developer make changes in system based on feedback & then the users & clients are again allowed to use the system.
- This cycle repeats until obtaining feed back is outweighed by cost & time.
- In prototyping model, only those features are included that will have a valuable return from the user experience.
- Exception handling, recovery & conformance to some standards & formats are typically not included in prototype.

Advantages:-

1) The experience of developing the prototype will reduce the cost of the actual software development.
2) Requirement will be more stable, due to feedback from prototype.
3) The quality of final software is likely to be far superior.
4) The prototyping is well suited for project where requirements are ~~not properly~~ hard to determine.

3) Iterative Development :-

- Iterative development process model counters the third & fourth limitation of waterfall model.

- Iterative Development model combines the benefits of both prototyping & the waterfall model.

- The basic idea is that the software should be developed in increaments, each increment adding some functional capability to the system. until the full system is implemented.

- The iterative enhancement model is an example of this approach.
- The 1st step of this model is make a subset of the overall problem.
- This subset is one that contains some of the key aspects of the problem

that are easy to understand & implement and which forms a useful & usable system.

- A project control list is created that contains, in order all the tasks that must be performed to obtain the final implementation.

- Each step consist of three phases named as, the design phase, implementation phase & analysis phase.

- The process is iterated until the project control list is empty.

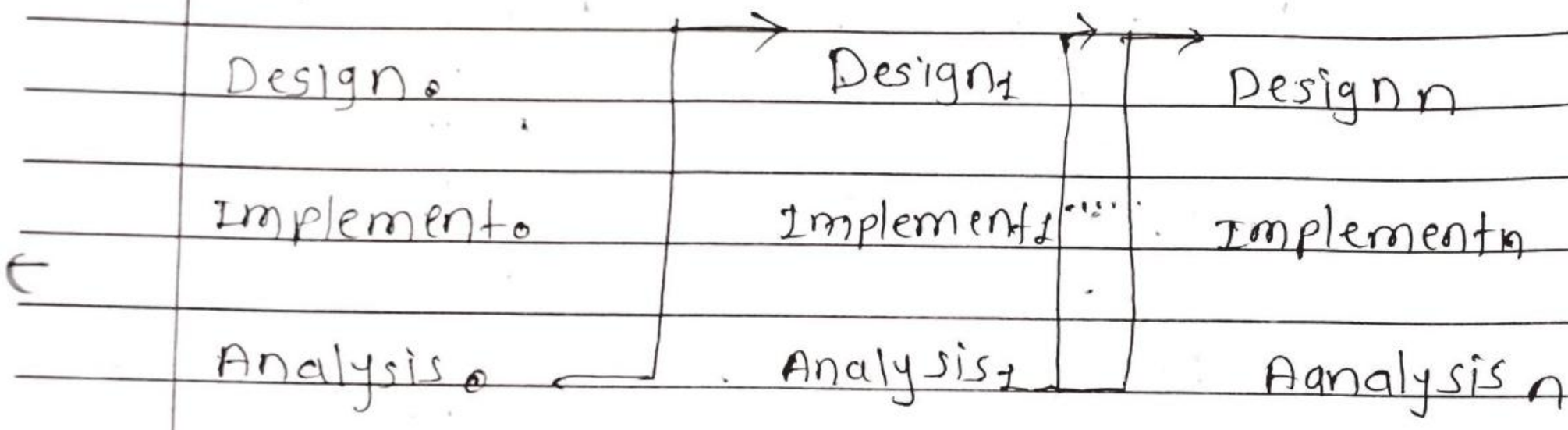



fig. The iterative enhancement model.

## * Iterative delivery approach:-

- Another common approach for e iterative development is to do the requirements and the architecture design in a standard waterfall or prototyping approach, but

deliver the software iteratively
- At the start of each delivery iteration, which requirements will be implemented in this release are decided, and then the design is enhanced & code developed to implement the requirement.
- The iteration ends with delivery of a working software system providing some value to end user.

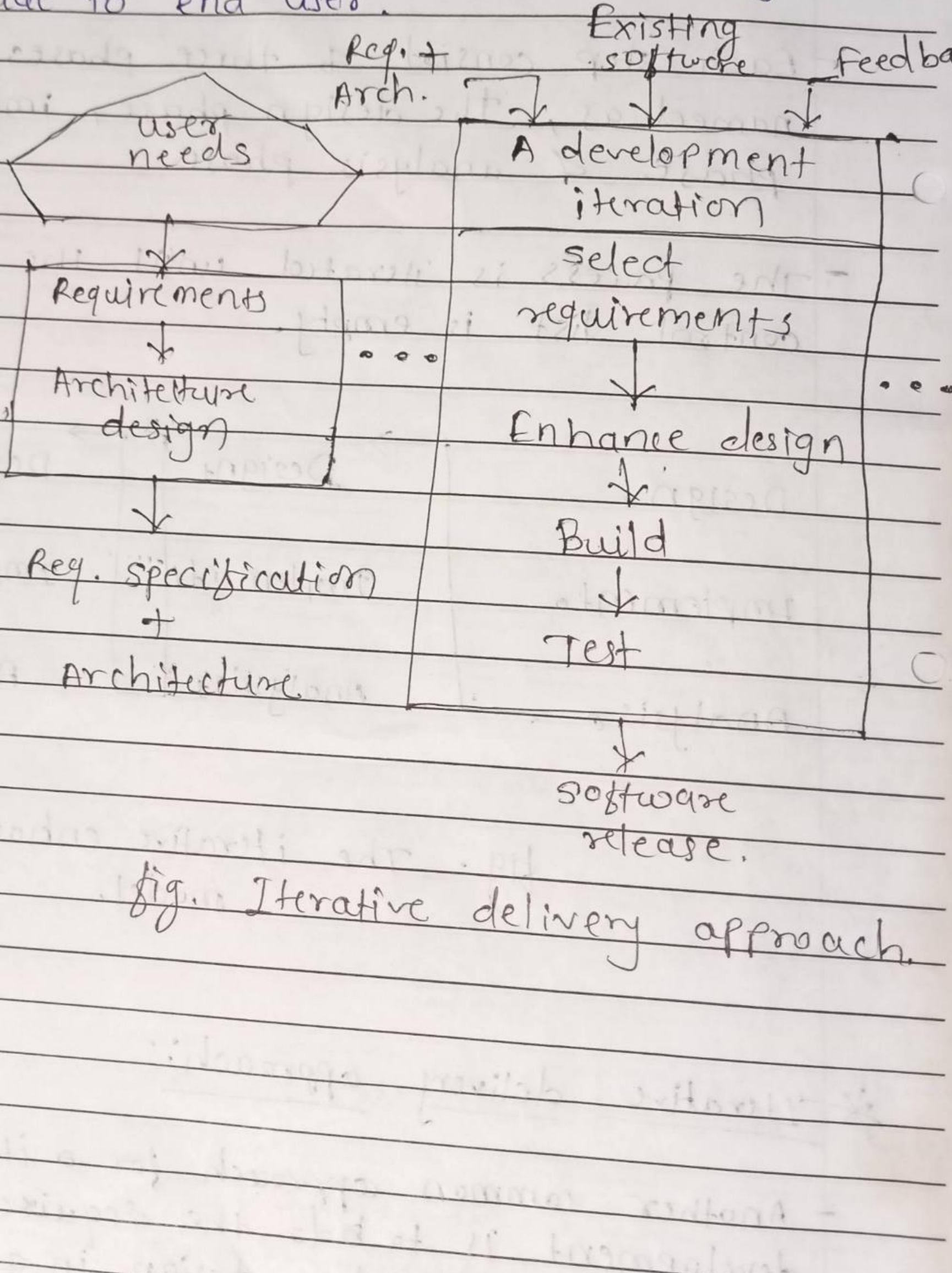



fig. Iterative delivery approach.

## * Rational Unified process:-

- designed by Rational, now part of IBM.
- It was designed for object-oriented development using the Unified modeling Language (UML)
- It proposes that development of s/w be divided into cycles
- Each cycle delivering a full working system.
- Generally each cycle is executed as a seperate project whose goal is to deliver some additional capabilities to an existing system.
- Each cycle itself is broken into following steps:-

(1) Inception phase
(2) Elaboration phase
(3) construction phase
(4) Transition phase.

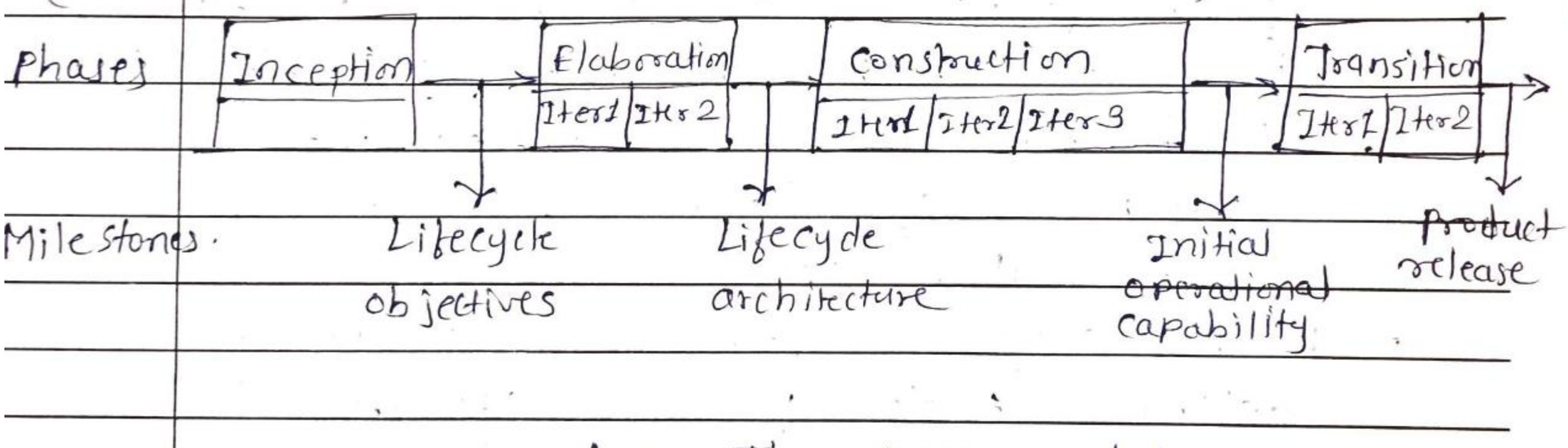



fig. The RUP model.

- The purpose of the inception phase is to establish the goals & scope of the project & completion of this phase is the lifecycle objectives milestone.

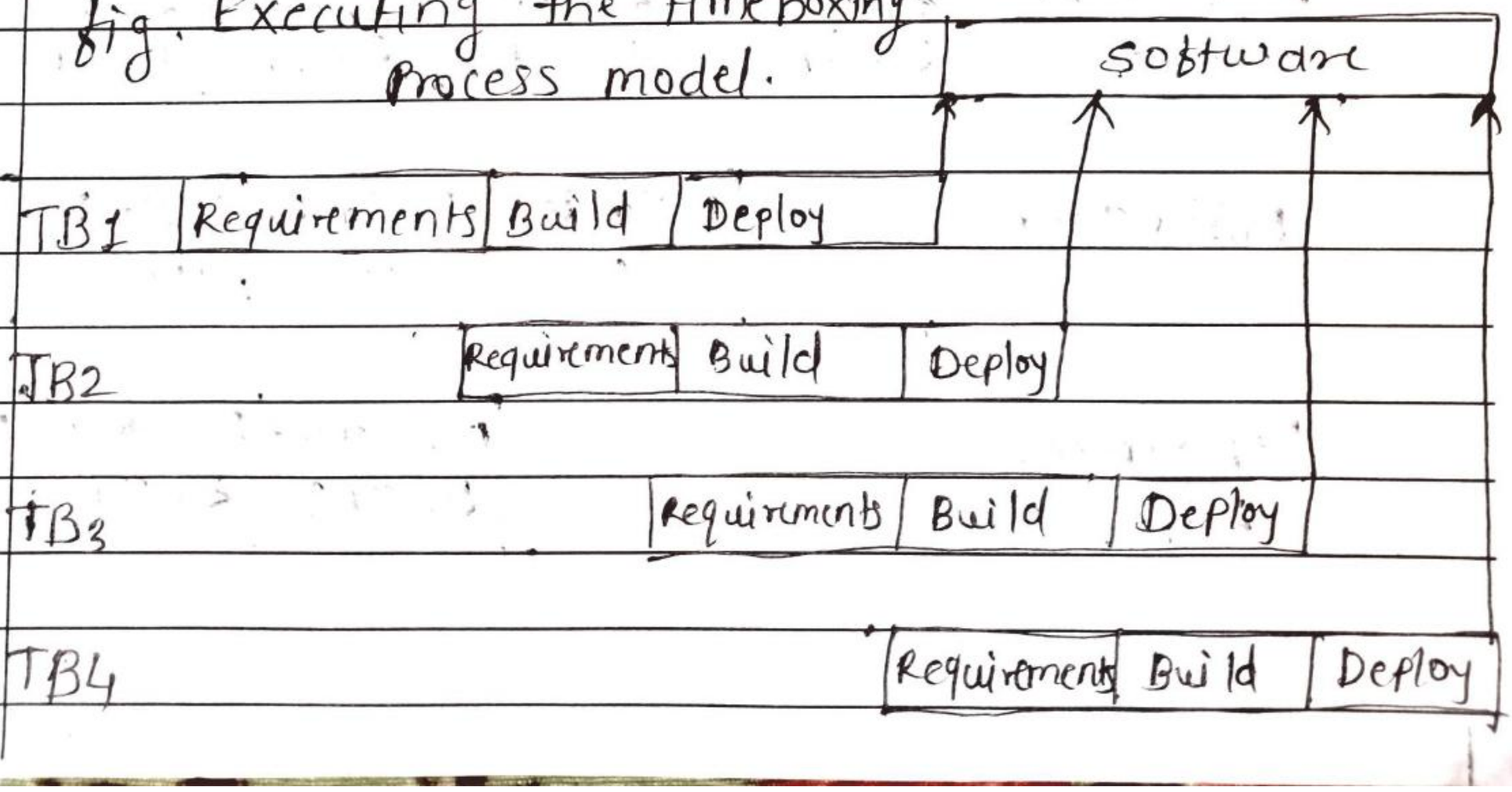
- In the elaboration phase, the architecture of the system is designed, based on the detailed requirement analysis
- At the end of this phase, it is expected that most of the requirements have been identified & specified.
- In the construction phase, the software is build & tested
- The purpose of the traditional phase is to move the software from the development environment to the client's environment.

## * Time boxing model:-

- In the timeboxing model, the basic unit of development is a time box, which is of fixed duration.
- Since the duration is fixed, a key factor is in selecting the requirements or features to be built in time box is what can be fit into the time box.
- 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 & produces a clearly defined output.
- The duration of each stage, that is the time it takes to complete the task of that stage; is approximately the same.
- The model requires that there be a dedicated team for each stage.

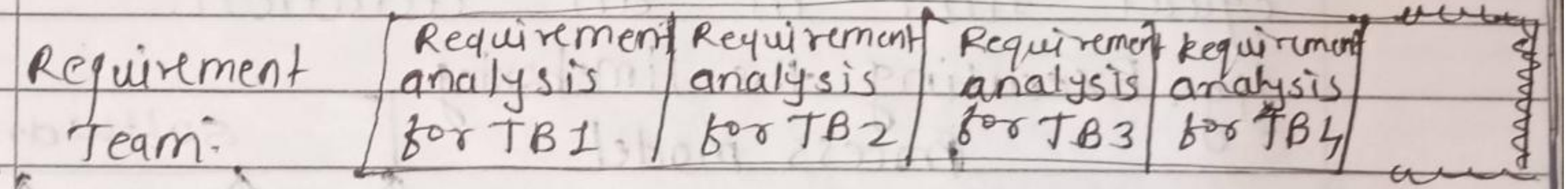
- The team for a stage performs only tasks of that stage - tasks for other stages are performed by their respective teams.
- To illustrate the use of this model, cosider a time box cosisting of three stages: requirement specification, build, and deployment.
- The requirement stage is executed by its team of analysts & ends with a prioritized list of requirements to be built in this iteration along with high-level design.
- The built team developes the code for implementing the requirements, and performs the testing.
- The tested code is then handed over to the deployment team which perfoms 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.

fig. Executing the timeboxing process model.



- 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 software.
- The requirements team then goes on & starts preparing the requirements for timebox 2.
- when the build for timebox 1 is completed, the code is handed over to the deployment team & the build team moves on to build code for requirements for timebox 2, and the requirements team moves on to doing requirements for timebox 3.
- This pipelined execution of the timeboxing process is shown in figure.

Fig:- Tasks of different teams.



## * Extreme programming of Agile processes.

- Agile approaches are based on some common principles:-

(1) working software is the key measure of progress in project.
(2) For progress in a project, therefore, software should be developed and delivered rapidly in small increments.
(3) Even late changes in the requirements should be entertained.
(4) Face-to-face communication is preferred over documentation.
(5) continuous feedback & involvement of customer is necessary for developing good quality software.
(6) simple design which evolves & improves with time is better approach than doing an elaborate design upfront for handling all possible scenarios.
(7) The delivery dates are decided by empowered teams of talented individuals.

## * Project management process:-

- The project management process specifies all activities that need to be done by the projed management to ensure that cost & quality objectives are met.
- its basic task is to plan the detailed implementation of the process for the perticular project. & then ensures 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 in to three phases:-
  i) planning.
  ii) monitoring & control.
  iii) Termination analysis.

i) project management begins with the planning, which is the most critical activity.

The goal of this phase is to develop a plan for software development following which the objectivs of project ~~toh~~ can be met successfully & efficiently

During planning, the major activities are cost estimation, schedule & milestone determination, project staffing, quality control plans, & controlling & monitoring plans.

ii) Project monitoring and control phase of the management process.
- It is longest in terms of duration.
- It encompasses most of the development process.
- It includes all the activities the project management has to perform.
- Monitoring potential risks for the project, which might prevent the project from meeting its objectives, is another important activity during this phase.
- The development process provides the information, the management process needs.

iii) Termination analysis :-
- The basic reason for performing termination analysis is to provide information about the development process. & learn from the project in order to improve the process.
- This phase is often called as postmortem analysis.
- In Iterative development, this analysis can be done after each iteration, to provide feedback to improve the execution of further iterations.