JEPPIAAR INSTITUTE OF TECHNOLOGY

"Self-Belief | Self Discipline | Self Respect"

DEPARTMENT

OF

COMPUTER SCIENCE AND ENGINEERING

LECTURE NOTES

IT8075 – Software Project Management

(Regulation 2017)

Year/Semester: IV/07 CSE

2021 – 2022

Prepared by

Ms. R. Revathi

Assistant Professor/CSE

UNIT-II Project life cycle and Effort Estimation

Software process and process Models - choice of process Models - Rapid Application development - Agile methods - Dynamic System Development Method - Extreme programming - Managing interactive processes - Basics of Software estimation - Effort and cost Estimation techniques - COSMIC Full function points - COCOMO II - a parametric Productivity Model

Software process and process Models

Software process

A software product development process usually starts when a request for the product is received from the customer.

The product depends on the type of company

It involves following stages. They are

i) Inception -

The Expression of need for a particular product is called Inception stage

ii) Maintenance -

In this stage, the series of Transformations which the product undergoes until it is fully developed is carried out.

The product needs to be maintained for fixing errors and adding new functionalities to the product

iii) Retirement -

When the developed product is no longer useful to the customer, it is retired.

- This set of stages through which a product undergoes from inception to retirement forms the product life-cycle

- This process is also called as 'software development life cycle (SDLC)" and s/w process.

Process Models

A process is "system in action"

* The process model of a software product is the graphical or textual form of the product's life cycle.

* It acts as a guide for developing a s/w product. It should provide the complete details about the various activities involved in developing a product

* The process model should describe about the different phases in product development (i.e.,) inception stage, maintenance and retirement stage

Choice of process Models

To develop a product, the system will have to execute one or more activities.

The activities can be organized in different ways.

There are no of process models available to develop a product. Some of them are

Waterfall Model
Spiral Model
RAID Model etc

All these methodologies can't be applied for developing a product.

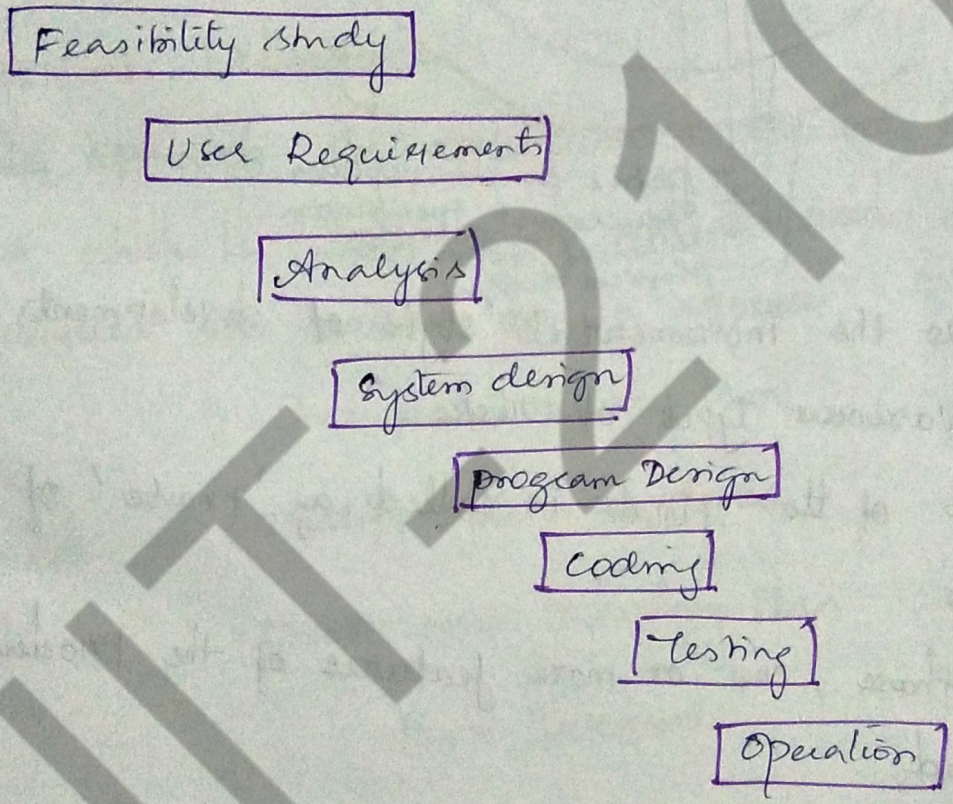
* The choice of a process model mainly depends on 2 factors

1. structure
2. Speed of delivery.

The choice of a process model also depends on the type of product we want to develop.

Waterfall Model

This is the 'classical' model of system development that is also known as 'one-shot' or 'once-through' model



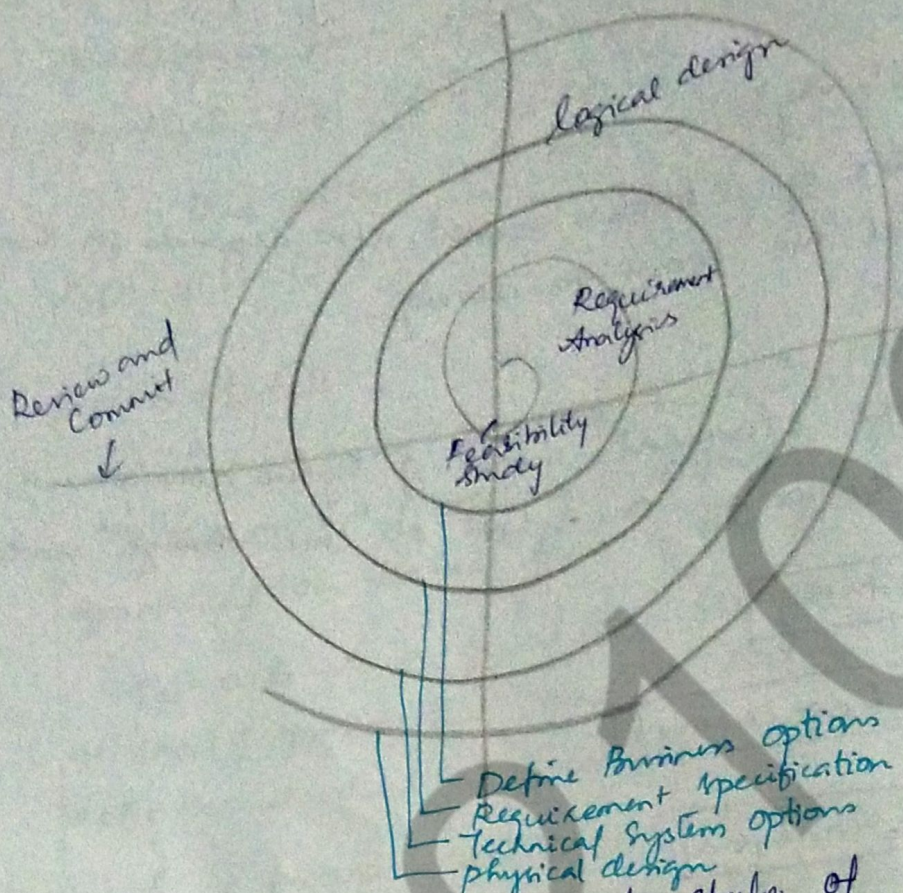
Adv.

* Easy & simple to understand
* Phases are processed & completed one at a time
* It has clearly defined strategy
* Tasks are easily arranged

Dis-Adv

* No s/w product will be produced until late during entire lifecycle is completed
* This model is not suitable for complex and OO projects
* This model cannot accomodate changing requirements

Spiral Model:



- It follows the incremental style of development and it handles various types of risks.
- Each loop of the spiral is called a 'phase' of s/w process.
- In each phase, one or more features of the product are implemented.

Advantages

- changing requirements can be accomodated
- Requirements can be captured accurately

Disadvantage

- management is more complex
- process is complex
- spiral may go indefinitely

Software Prototyping:

- A prototype is a working model of one or more aspects of the projected system.

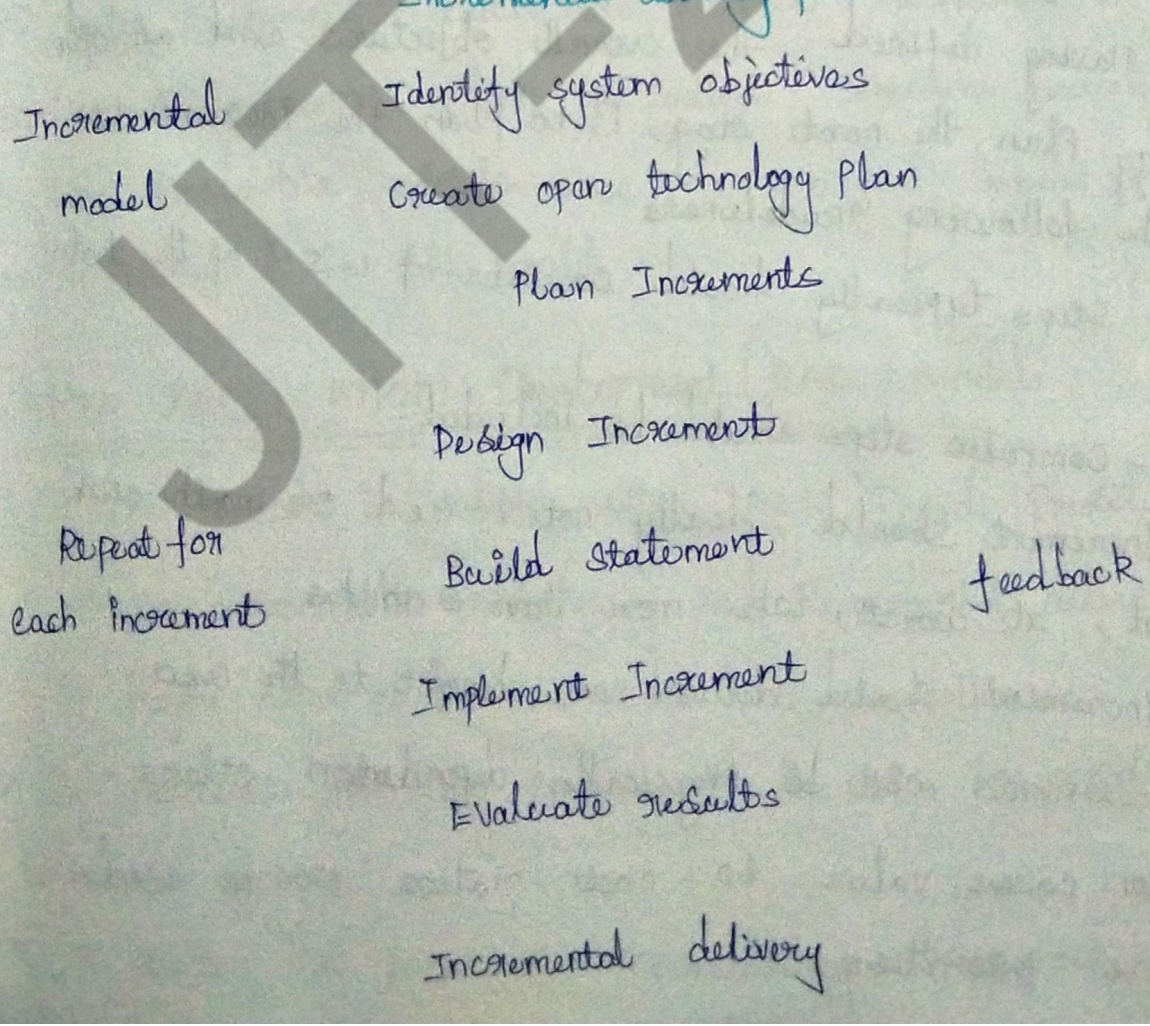
- It is constructed and tested quickly and inexpensively over assumptions.
- prototypes can be classified as throco-away or evolutionary

Throco-Away prototype → The prototype tests out some ideas and is then discarded when the true development of the operational system is commenced.

Evolutionary Prototypes → The prototype is developed and modified until it is finally in a state where it can become the operational system.

INCREMENTAL Delivery :-

- In this approach, the application is divided into smaller components which is implementing and delivered in sequence
- The activities are carried out in sequence.



Incremental Delivery Plan :-

- All the nature and order of every increment is Planned at the beginning, which is similar to Strategic Planning.

The key elements of the phases are,

1. System objectives
2. Incremental plan
3. open technology plan

1. System objectives :-

- In this step, the objective should be well defined.

It is subdivided into :

- Intended objective
- Tasks which system has to do.
- Details about computer / non- computer functions

2. Incremental plan :-

- Having defined the overall objectives and an open technology plan, the next stage is to plan the increments using the following guidelines

* Steps typically should consists of 1-5 % of the total project
* Non - Computer steps should be included.
* An increment should, ideally not exceed one month and should not, at worst, take more than 3 months.
* Each increment should deliver some benefit to the user.
* Some increments will be physically dependent on others.
* In other cases value - to - cost ratios may be used to decide priorities.

3. Open technology Plan :-

- This plan includes the following

* A standard high-level language.
* A standard operating system.
* Small modules.
* Variable parameters in a file.
* A standard database management system.

Advantage :-

-> The feedback from early increments will improve later steps
-> Smaller components are simpler to control and manage.
-> Unnecessary features for current increment can be omitted and added in the next, which is called "Global Plating".

DisAdvantage :-

-> later increments which are added requires changes to earlier increments, which is called "software breakage".
-> Less Productive, since, a series of smaller processes are implemented in this model.

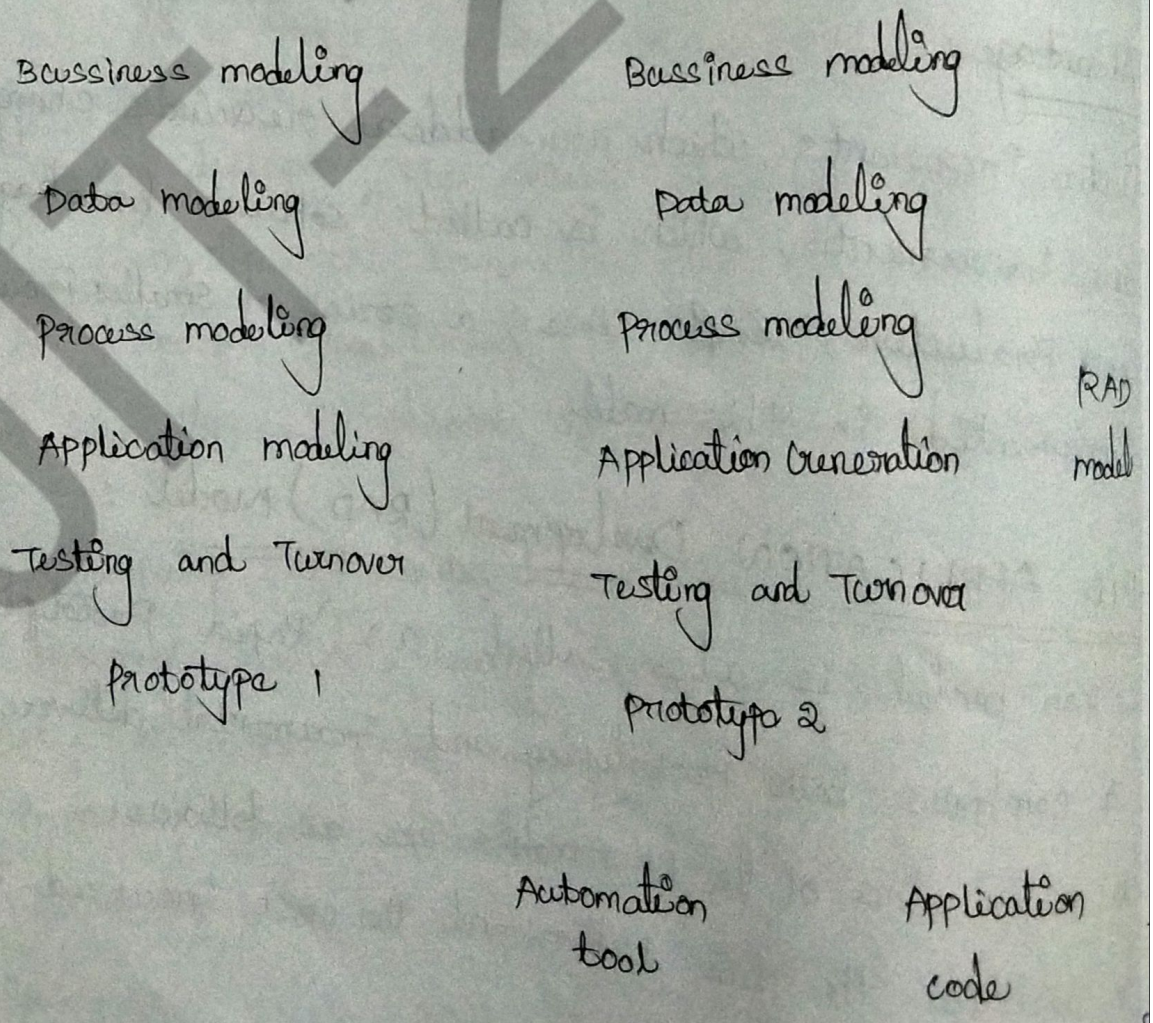
RAPID APPLICATION Development (RAD) model :-

- RAD model is also called as "Rapid Prototyping model".
- It combines both prototyping and Incremental delivery
- The major aims of the RAD models are as follows :-

* To decrease the time taken and the cost incurred to develop software systems.
* To limit the costs of accomodating change requests by

incorporating them as early as possible before large investments have been made in development and testing.

- In RAD model, the development takes place into a series of short cycles called "Iterations".
- plans are made for one iteration at a particular time.
- The time planned for each iteration is called as "time box".
- Each iterations enhances the implemented functionality and a Quick prototype is developed.
- The prototype is given to the customer, then the customer evaluates it and provides the feedback.
- This is repeated for each iterations and over successive iteration, the prototype takes a final shape.



- RAD team uses specialized automation tools for faster creation of working prototype
- In the above diagram, the prototypes are created for each and every iterations.
- RAD model also has a customer representatives to clarify about the requirements.

AGILE METHODS

- Agile methods are designed to overcome the disadvantages of the traditional implementation methods.
- "An Agile model is an umbrella term that refers to a group of development processes".
- There are various agile approaches such as the following.
  * Crystal Technologies
  * Atern (formerly DSDM)
  * Feature-driven development
  * Scrum
  * Extreme Programming (XP)

Features of Agile methods:

-> In agile methods, the feature requirements are divided in several small parts. Each part is developed in an iteration
-> Each iteration is taken as a easily manageable, short term plan. The time taken to complete an iteration is called "time-box".
-> Agile model uses face-to-face communication over written documents. The teams size is small and it consists of only 5-9 members, which provides effective communication.

⇒ Contacts between team members may be done through e-mail video conferencing, telephone, etc.

→ In agile method, a customer representation is present to review the progress made, re-evaluate the requirements and to provide suitable feedback to the development team.

⇒ Agile methods usually follows pair programming (ie) Here, two programmers work together at one workstation one person types the code and other person reviews it. The two persons can change their role for every hour or so

⇒ This helps to reduce the errors of the various agile methods, the most commonly used are "Extreme Programming" and "SCRUM"

Extreme Programming :.

- Kent Back's Extreme Programming is first Published in 1999 and updated in 2004.
- The idea is called "extreme Programming" Because, according to Back, "XP takes commonsense Principles to of XP.
- Four core values are Presented as the foundations of XP

1. Communication and Feedback :-

→ The best method of communication is face to face method.

→ For documentation is avoided.

2. Simplicity :-

-> implements the user's requirements should be done in a simpler design complex methods should be avoided.

3. Responsibility :-

- The developers are solely responsible for the Quality of the software.

4. Courage :-

- Trying out new ideas and if they don't work out they should be scrapped.

Core Practices of XP :-

* Planning exercise :-

- In XP, code is developed in iterations, Periods of one to 4 weeks duration, during which specific features of the s/w are created. These are called as "releases".

- The planning exercises is a process, where the features to be incorporated in next release are negotiated.

* Small Releases :-

- The time between releases of functionality to the users should be short (ie) it should be a months or two.

* Metaphor :-

- The System to be built will be s/w code, that reflects things that exist and happen in the real world.

- eg. howdy-rate, Calculates-gross-Pay.

- 'architecture' refers to the use of system models such as class and collaboration diagrams to describe the system. 'Architecture' is itself a metaphor.

* Simple design :-

- Practical implementation of the value of simplicity that was described above.

* Testing :-

- Testing is done at the same time as coding.
- It should be done to check whether the expected results arrive for the test inputs.
- Testing is carried out using an automated testing
- There are 2 types of testing normally used.

1, UNIT Testing -> which is used to focus the code written by developer.

2, FUNCTION Testing -> which is user-organized and checks the correctness of a particular feature.

* Refactoring :-

- modifying the part of code as a result of coding some changes is called "refactoring"
- we have to ensure that no bug has been introduced due to refactoring.

* Pair Programming :-

- All software code is written by pairs of developers, one actually doing the typing and others observing.

- This is used to reduce errors and improve performance.

* Collective ownership :-

- This is really the corollary of pair programming.

- The team as whole takes the collective responsibility for the code in the system.

* Continuous integration :-

- this is another aspects of testing practices.

- As changes are made to sw units, integrated tests can be run regularly to ensure correctness of components.

* Forty - hour weeks :-

- It points at that working excessive hours can lead to ill-health and be generally counter productive.

The principles is that normally developers should not worose more than 40 hours a week.

* On-Site Customers :-

- Fast and effective communications with the users is achieved by having a user domain

export on-site with the developers.

* Coding standards :-

- If code is genuinely to be shared, then there must be common, accepted, coding statements to support the understanding and ease of modification of the code.

Limitations of XP :-

The successful use of XP is based on certain condition. If these do not exist, then its practice can be difficult. These conditions includes the following

* There must be easy access to users.
* Development staff need to be physically located in the same office.
* Large complex systems may initially need significant architectural effort.

SCRUM

-> In this model, projects are divided into small units of work.
-> These are delivered over the boxes which are called "sprints".
-> At the end of each sprint, the progress of the project is analysed and suggestions are

given to make improvements.

Product backlog → Sprint Planning → Sprint backlog → Product

Scrum Process

- members in a scrum Process

There are 3 vital members in a scrum model. They are.

i, owner

ii, Scrum master

iii, Team member

i, Owner :- Creates a wish list called a Product backlog

- The product owner communicates the requirements of the customer to the development Plan.

ii, Scrum master :- It keeps the team focused on its goal and acts as a interface between owner and team.

iii, Team member :- It is responsible for development of the project according to the backlog.

Advantages of Scrum model :-

1, used to implement complex Projects.

2, improve the team work and communication.

3, productivity can be improved with daily meetings.

4, Product can be delivered in a scheduled time.

Disadvantages of Scrum model:-

- If the task is not well-defined, Sprint Process will take much time.
- Team members should be well committed to the task. If they fail, project will also fail.
- In-experienced team members should not be able to complete Project in time.
- Regression testing should be conducted after each Sprint to implement Quality management.

Hence, the scrum model can be used in situations where the team members should be experienced and committed.

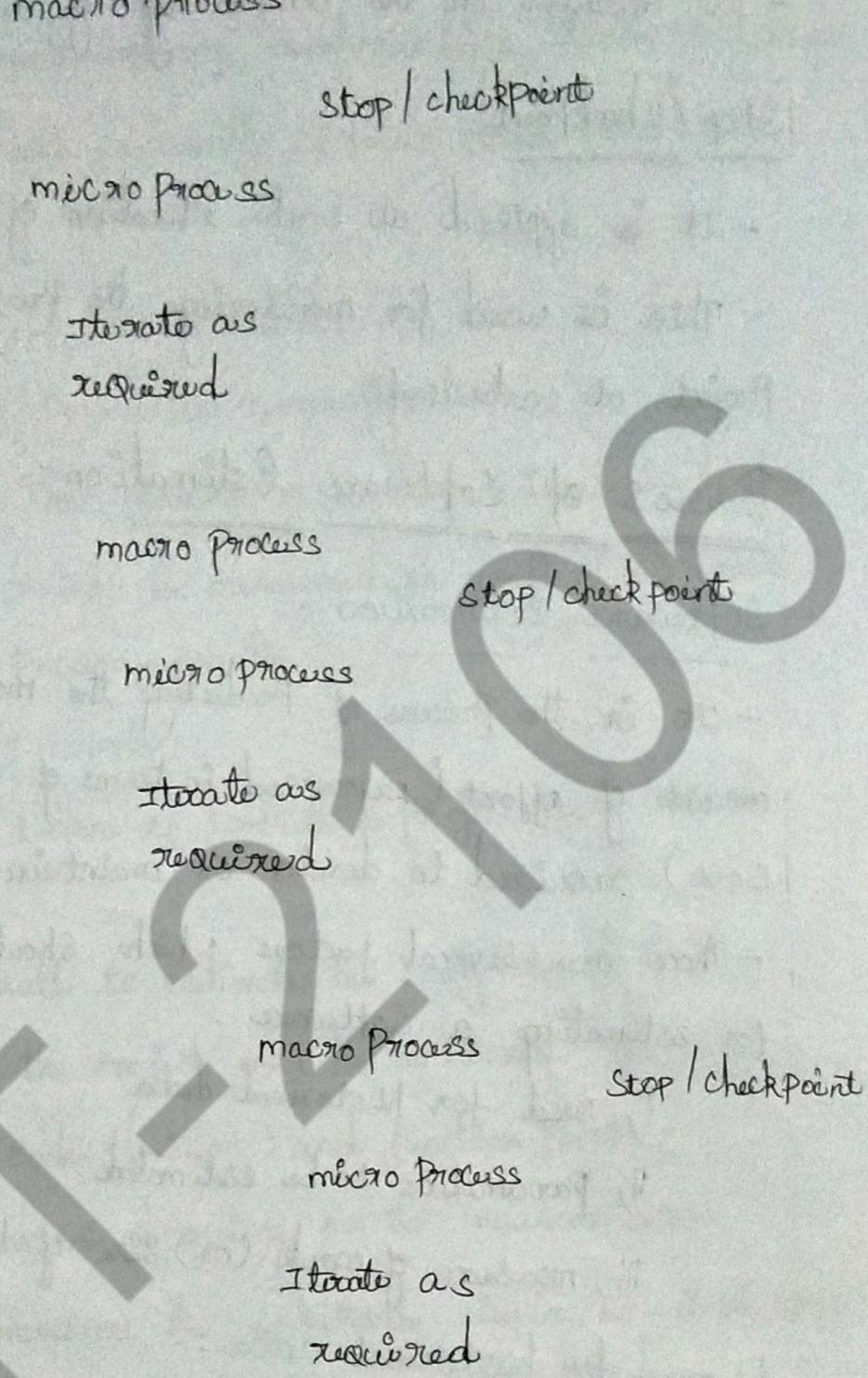
Managing Iterative Processes:-

- Booch suggest 2 levels of development for managing iterative Process. They are.

1, Macro Process
2, Micro Process

Macro Process:-

- It is similar to waterfall model.
- The activities for the concerned groups should be co-ordinated.
- The dates for that activities will be known after the completion of activities, Subsequent activities have to be Performed.



A macro process containing 3 iterative micro processes.

Micro Process:-

- The microprocesses are defined inside macro processing. These processes are divided into several iterations.
- Time-boxes are used to control the processes.
- In some cases, the macro process itself can be iterative.

- It depends on the complexity of a Project.

Stop / CheckPoint :-

- It is applied at each iteration of the micro Proc
- This is used for monitoring the Progress of the Project at each level.

Basics of Software Estimation :-

Software Estimation :

- It is the process of predicting the most realistic amount of effort (expressed in terms of Person-hours or cost) required to develop or maintain a Software.
- There are several factors which should be considered for estimating a Software.
  i) Need for historical data
  ii) Parameters to be estimated
  iii) measure of work (or) size of the project.

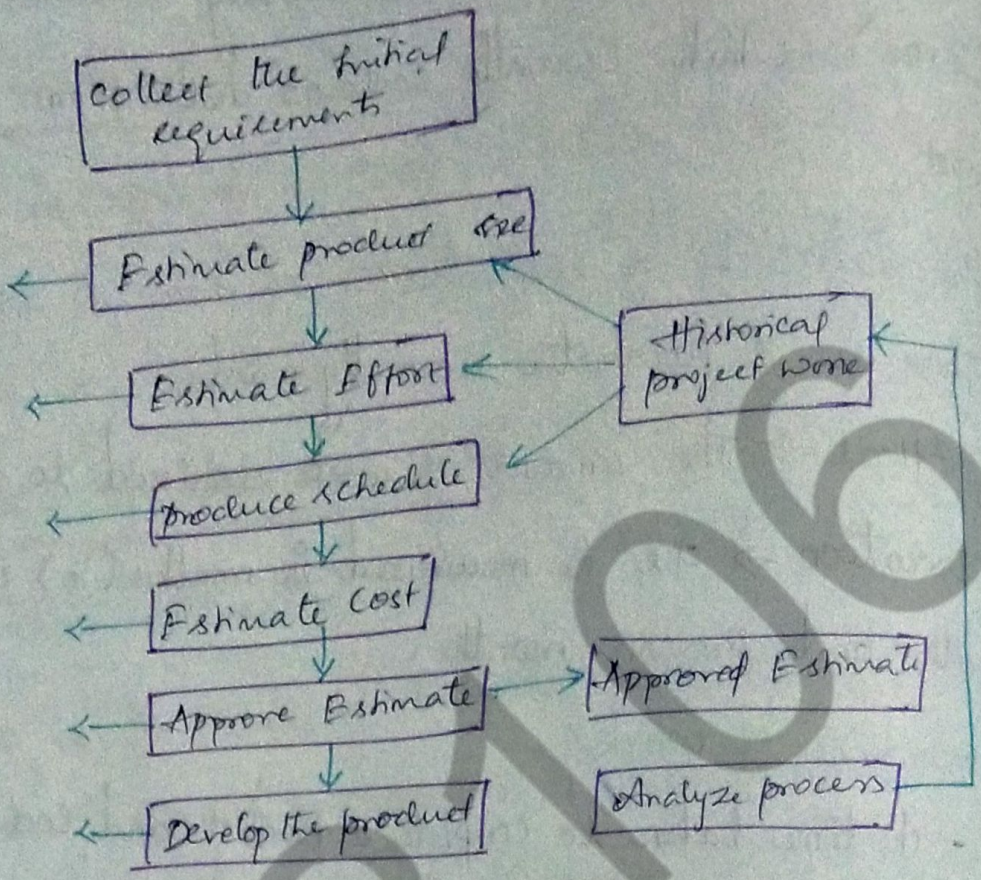
i) Need for historical data :-

- Past projects data can be used for estimation. This past data cannot be applicable in all situation, because of the emergence of new Programming languages and methods.
- External data sets can also be used, some where, according to the Project.

eg:-
International Software Benchmarking Standard group (ISBE), which currently contains data from 4800 Project.

ii) Parameters to be estimated:-
- There are 2 parameter normally used
a) Effort => The amount of work required to develop
b) Duration -> It is measured in months (ie) work-month and person-month.

iii) Measure of work:-
- The time taken to complete a project and cost is estimated.
- It is difficult to estimate the early stages of planning.
- The size of the project may be measured in source lines of the code (SLOC) and function point (FP)
- The size varies depending on the measure used.
- The SLOC measure is intuitively simple, so it is still being widely used. It is important, however, to be aware of the major short comings of the SLOC measures.
  * No precise definition
  * Difficult to estimate at start of project
  * Only a code measure.
  * Programmer-dependent
  * Does not consider code complexity.



Estimation process

Effort and Cost Estimation Techniques

Barry Boehm, identified the main ways of deriving estimates of s/w development effort as

Algorithmic models

which use 'effort drivers' representing characteristics of the target system and the implementation environment to predict effort

Expert judgement based on the advice of knowledgeable staff

Analogy where a similar, completed, project is identified and its actual effort is used as the basis of the estimate

parkinson

where the staff effort available to do a project becomes the estimate

price to win

where the estimate is a figure that seems sufficiently low to win a contract

top-down where an overall estimate for the whole project is broken down into the effort required for component tasks

Bottom-up where component tasks are identified and their individual estimates are aggregated

Bottom up Estimating

* The Estimator breaks the project into its component tasks with a large project, the process of breaking it down into tasks is iterative.

* Each task is decomposed into its component subtasks and these in turn could be further analysed

* The Bottom-up approach is best at the later, more detailed, stages of project planning If this method is used earlier, assumptions about the characteristics of the final system and project work methods will have to be made.

* When a project is completely novel or there is no historical data available, the estimator would be forced to use the bottom up approach

A procedural code-oriented approach

We describe how a 'bottom up approach can be used at the level of s/w components.

* Envisage the no. and type of s/w modules in the final system.

* Estimate the SLOC of each identified module

* Estimate the work content, taking into account complexity and technical difficulty.

* calculate the work days effort

eg: Cocomo model

Top-down approach and Parametric models:-

→ also called macro model.

→ Top-down approach is normally associated with Parametric (or algorithmic) models.

- example for Parametric model is estimating the cost of rebuilding a house.

- Project effort relates mainly to variables associated with characteristics of the final system. A Parametric model will manage normally have one or more formulae in the form

effort = (system size) x (productivity rate)

productivity = effort / size

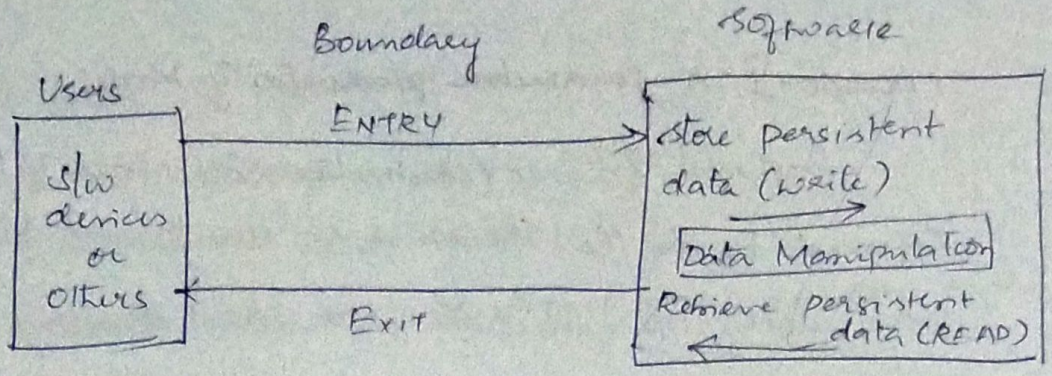
e.g:- Putnam model

COSMIC FULL FUNCTION POINTS

- IFPUG (International FP user Group) approach is not suitable for real-time (or) embedded applications. Hence, another version called "COSMIC method" is developed, COSMIC FFP method (Full Function Point).

- Cosmic decompose the system Architecture in to s/w layers and process them. The s/w components to be sized can receive results for services from layers above.

- The communication followed is "peer to peer communication"



COSMIC FFP Model

Data groups can be moved about in 4 ways

Entries(E) - which are effected by subprocesses that move the data group into the s/w component in question from a 'user' outside its boundary. This could be from another layer or another separate s/w component in the same layer via peer to peer communication

Exits(X) - which are effected by subprocesses that move the data group from the s/w component to a 'user' outside its boundary

Reads(R) - which are data movements that move data groups from persistent storage into the s/w component

Writes(W) - which are data movements that transfer data groups from the s/w component into persistent storage.

Overall FFP count is derived by simply adding up the counts for each of the 4 types of data movement. The resulting units are Cfsu (cosmic functional size units).

COSMIC FFPs have been incorporated into an ISO standard - ISO/IEC 19761:2003

COCOMO-II : A parametric productivity Model

Boehm's 'COCOMO (Constructive Cost Model) is often Referred to in the literature on s/w project Management particularly in connection with s/w Estimating.

- The term cocomo really refers to a group of models,

Types of models.

1. Organic
   - The system developed will be small
   - The environment will be highly familiar
   - The interface requirements will be flexible
2. Embedded
   - The constraints will be tight
3. Semi-detached
   - Combination of organic and embedded models.

- stages in cocomo model

1. Application composition
   - External features of the system will be designed and prototyping will be implemented
2. Early design – Architecture of the system will be adapted
   - Fundamental s/w structures are designed
3. Post Architecture - The designed s/w structures will undergo final construction and modification. Finally, a system will be created as required

To calculate an estimate of person months

$$Pm = A(Size)^{(Sf)} \times (em_1) \times (em_2) \times \cdots (em_n)$$

Where A = Constant

Size = it is measured in kdsi

Sf = scale factor

The scale factor is calculated as

$SF = B + 0.01 \times \sum$ (exponent driver ratings)

Here Bias constant which is 0.91

- scalefactor value vary for project types, according to the size of the project

Driver	Very low	low	Normal	High	Very high	Extra high
PREC	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	5.07	4.05	3.04	2.03	1.01	0.00

COCOMO II Scale factor values

The effort multipliers (em) adjust the estimate to take account of productivity factors, but do not involve economies or diseconomies of scale

eg:

code	Effor modifier	Extra low	Very low	low	Normal	High	Very high	Extra high
RCPX	product reliability and complexity	0.49	0.60	0.83	1.00	1.33	1.91	2.72
~~RUSE~~				0.95	1.00	1.07	1.15	1.24
RUSE	Required reusability							

COCOMO II - Early design effort multipliers

Modifier Type	code	Effort modify
Product attributes	RELY	Required s/w reliability
	DATA	DB size
	DOCU	Documentation match to life cycle needs
	CPLX	product complexity
	REUSE	Required reusability

COCOMO II post Architecture effort multipliers

Staffing Pattern

- staffing requirement for the particular project can be determined, after the estimation is done.
- Staffing pattern for sw projects was studies by 2 persons. They are

Putnam
Norden

Putnam was first to study the problem for the staffing pattern and he also extended the work of Norden

Norden and putnam's work are the two vital staffing pattern desirable to study

Norden's work

Norden studied staffing patterns of Research & development projects (R&D projects)

Now, the staffing pattern of R&D projects is different from manufacturing or sales type of work.

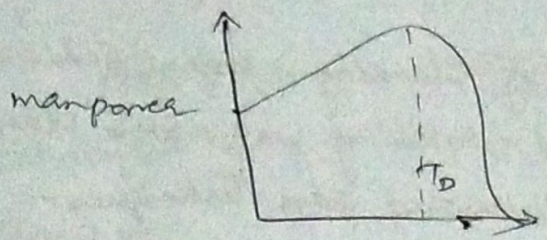
Eg: In a supermarket, the no. of sales persons would depend on the no. of sales counters and the no. of sales persons remains fixed for years together

In the case of R&D project, staffing pattern will change dynamically over time

This is because of 2 reasons

1. When a R&D project is started, the activities of project are planned as per the first phase. considering this, the manpower requirement are low.
2. As the project enters into subsequent phases, the requirements for staff will gradually increase overtime

Therefore, Norden concluded that the staffing pattern for any project of this kind can be shown by a curved called as "Rayleigh distribution curve"



Rayleigh curve

Putnam's work

It focused on the staffing pattern of the sfw development project. He found out that the staffing pattern of sfw development project were similar to R&D projects

Putnam considered 2 factors regarding the sfw development. They are

1. Lines of code
2. Time required to develop the product

During the start up of a project, small no's of developers are needed only to carry out planning & specification tasks.

As the project progresses, the no of people will increase and reaches a peak, which is shown in Rayleigh curve

After the product is delivered, the staff count will decrease during maintenance

Putnam concluded that staff building up should not be carried out in large installments and staff reduction should be done gradually