Software Project Management

By
Anil Kumar Dudyala
Source
(Rajib Mall)

Outline of the Lecture:

- Introduction to Project Planning
- Software Cost Estimation
  - Cost Estimation Models
  - Software Size Metrics
  - Empirical Estimation
  - Heuristic Estimation
  - COCOMO
- Staffing Level Estimation
- Effect of Schedule Compression on Cost
- Summary

Introduction

- Many software projects fail:
  - due to faulty project management practices:
    - It is important to learn different aspects of software project management.

Introduction

- Goal of software project management:
  - enable a group of engineers to work efficiently towards successful completion of a software project.

Responsibility of project managers

- Project proposal writing,
- Project cost estimation,
- Scheduling,
- Project staffing,
- Project monitoring and control,
- Software configuration management,
- Risk management,
- Managerial report writing and presentations, etc.

Responsibility of project managers

- A project manager's activities are varied.
  - can be broadly classified into:
    - project planning,
    - project monitoring and control activities.

Project Planning

- Once a project is found to be feasible,
  - project managers undertake project planning.

Project Planning Activities

- Estimation:
  - Effort, cost, resource, and project duration
- Project scheduling:
- Staff organization:
  - staffing plans
- Risk handling:
  - identification, analysis, and abatement procedures
- Miscellaneous plans:
  - quality assurance plan, configuration management plan, etc.

Project planning

- Requires utmost care and attention --- commitments to unrealistic time and resource estimates result in:
    - irritating delays.
    - customer dissatisfaction
    - adverse affect on team morale
        - poor quality work
    - project failure.

Sliding Window Planning

- Involves project planning over several stages:
  - protects managers from making big commitments too early.
  - More information becomes available as project progresses.
    - Facilitates accurate planning

SPMP Document

- After planning is complete:
  - Document the plans:
  - in a Software Project Management Plan(SPMP) document.

Imp

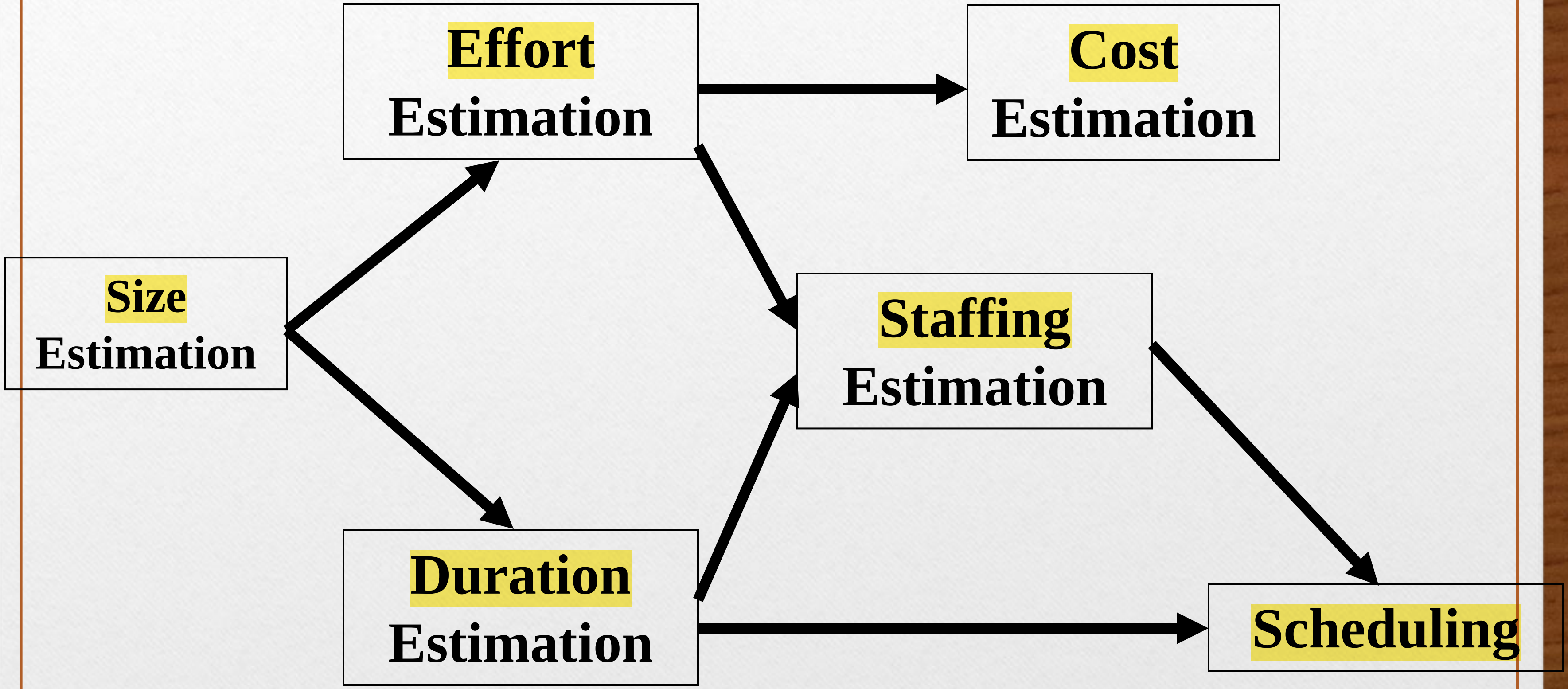
Organization of SPMP Document

- Introduction (Objectives,Major Functions,Performance Issues,Management and Technical Constraints)
- Project Estimates (Historical Data,Estimation Techniques,Effort, Cost, and Project Duration Estimates)
- Project Resources Plan (People,Hardware and Software,Special Resources)
- Schedules (Work Breakdown Structure,Task Network, Gantt Chart Representation,PERT Chart Representation)
- Risk Management Plan (Risk Analysis,Risk Identification,Risk Estimation, Abatement Procedures)
- Project Tracking and Control Plan
- Miscellaneous Plans(Process Tailoring,Quality Assurance)

Software Cost Estimation

- Determine size of the product.
- From the size estimate,
  - determine the effort needed.
- From the effort estimate,
  - determine project duration, and cost.

Software Cost Estimation



Software Size Metrics

- LOC (Lines of Code):
  - Simplest and most widely used metric.
  - Comments and blank lines should not be counted.

Disadvantages of Using LOC

- Size can vary with coding style.
- Focuses on coding activity alone.
- Correlates poorly with quality and efficiency of code.
- Penalizes higher level programming languages, code reuse, etc.

Disadvantages of Using LOC (cont...)

- Measures lexical/textual complexity only.
    - does not address the issues of structural or logical complexity.
- Difficult to estimate LOC from problem description.
    - So not useful for project planning

Function Point Metric

- Overcomes some of the shortcomings of the LOC metric
- Proposed by Albrecht in early 80's:
  - UFP=4 x #inputs + 5x #Outputs + 4x #inquiries + 10x #files + 10x #interfaces ?
  - FP = UFP x TCF
- UFP is further refined using the refinement of function point table.

Refinement of function point entities table

Type	Simple	Average	Complex
Input	3	4	6
Output	4	5	7
Inquiry	3	4	6
Number of Files	7	10	15
Number of interfaces	5	7	10

Function Point Metric

- Albrecht identified 14 parameters that influence the development effort.
- These 14 parameters are assigned a value from 0 to 6.
- The resulting sum is called the degree of influence.
- TCF = (0.65 +0.01*DI) DI would be in a range of 0.65 to 1.35.
- Finally, FP is computed as UFP*TCF.

Function Point Metric

- Input:
  - A set of related inputs is counted as one input.
- Output:
  - A set of related outputs is counted as one output.
- Inquiries:
  - Each user query type is counted.

Function Point Metric

- Files:
    - Files are logically related data and thus can be data structures or physical files.
- Interface:
    - Data transfer to other external systems.

Function Point Metric (CONT.)

- Suffers from a major drawback:
  - the size of a function is considered to be independent of its complexity.
- Extend function point metric:
  - Feature Point metric:
  - considers an extra parameter:
    - Algorithm Complexity.

Function Point Metric (CONT.)

- Proponents claim:
  - FP is language independent.
  - Size can be easily derived from problem description
- Opponents claim:
  - it is subjective --- Different people can come up with different estimates for the same problem.

Software Cost Estimation

- Three main approaches to estimation:
  - Empirical
  - Heuristic
  - Analytical

Imp

Software Cost Estimation Techniques

- Empirical techniques:
  - an educated guess based on past experience.
- Heuristic techniques:
  - assume that the characteristics to be estimated can be expressed in terms of some mathematical expression.
- Analytical techniques:
  - derive the required results starting from certain simple assumptions.

Empirical Size Estimation Techniques

- Expert Judgement:
  - An euphemism for guess made by an expert.
  - Suffers from Human error and individual bias.
- Delphi Estimation:
  - overcomes some of the problems of expert judgement.

Expert judgement

- Experts divide a software product into component units:
    - e.g. GUI, database module, data communication module, billing module, etc.

- Add up the guesses for each of the components.

Delphi Estimation:

- Team of Experts and a coordinator.
- Experts carry out estimation independently:
  - mention the rationale behind their estimation.
  - coordinator notes down any extraordinary rationale:
    - circulates among experts.

Delphi Estimation:

- Experts re-estimate.
- Experts never meet each other to discuss their viewpoints.

Heuristic Estimation Techniques

- **Single Variable Model:**
  - Parameter to be Estimated=C1(Estimated Characteristic)d1
  - $Estimated\ Parameter = c_1 * e^{d1}$
- **Multivariable Model:**
  - Assumes that the parameter to be estimated depends on more than one characteristic.
  - Parameter to be Estimated=C1(Estimated Characteristic)d1+ C2(Estimated Characteristic)d2+…
  - $Estimated\ resource = c_1 * ep^{d1}{}_1 + c_2 * ep^{d2}{}_2 + ...$
  - Usually more accurate than single variable models.

COCOMO Model

cost estimation - time + no of people

- COCOMO (COnstructive COst MOdel) proposed by Boehm.

- Divides software product developments into 3 categories:
  - Organic
  - Semidetached
  - Embedded

COCOMO Product classes

- Roughly correspond to:
  - application, utility and system programs respectively.
    - Data processing and scientific programs are considered to be application programs.
    - Compilers, linkers, editors, etc., are utility programs.
    - Operating systems and real-time system programs, etc. are system programs.

Elaboration of Product classes

- Organic:
    - Relatively small groups
        - working to develop well-understood applications.
- Semidetached:
    - Project team consists of a mixture of experienced and inexperienced staff.
- Embedded:
    - The software is strongly coupled to complex hardware, or real-time systems.

COCOMO Model (CONT.)

- For each of the three product categories:
  - From size estimation (in KLOC), Boehm provides equations to predict:
    - project duration in months
    - effort in programmer-months
- Boehm obtained these equations:
  - examined historical data collected from a large number of actual projects.

COCOMO Model (CONT.)

- Software cost estimation is done through three stages:
  - Basic COCOMO,
  - Intermediate COCOMO,
  - Complete COCOMO.

Basic COCOMO Model (CONT.)

- Gives only an approximate estimation:
  - Effort = a1 x $(KLOC)^{a2}$ PM
  - Tdev = b1 x $(Effort)^{b2}$ Months
    - KLOC is the estimated kilo lines of source code,
    - a1, a2, b1, b2 are constants for different categories of software products,
    - Tdev is the estimated time to develop the software in months,
    - Effort estimation is obtained in terms of person months (PMs).

Development Effort Estimation

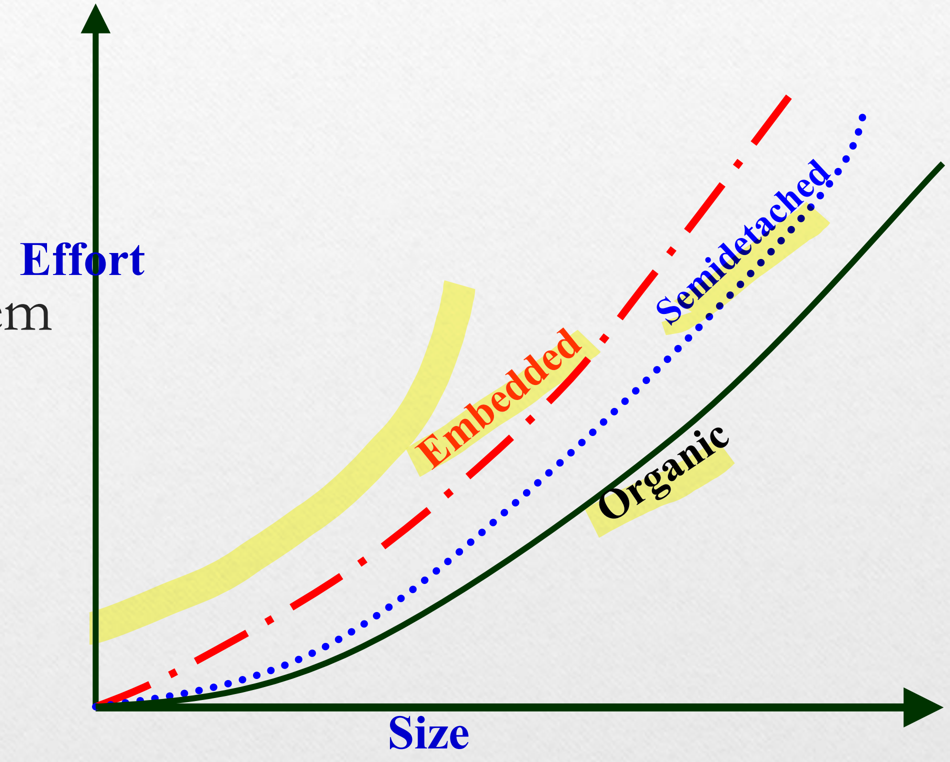
- Organic :
  - Effort = $2.4 (KLOC)^{1.05}$ PM
- Semi-detached:
  - Effort = $3.0(KLOC)^{1.12}$ PM
- Embedded:
  - Effort = $3.6 (KLOC)^{1.20}$ PM

Development Time Estimation

- Organic:
  - $Tdev = 2.5\ (Effort)^{0.38}$ Months
- Semi-detached:
  - $Tdev = 2.5\ (Effort)^{0.35}$ Months
- Embedded:
  - $Tdev = 2.5\ (Effort)^{0.32}$ Months

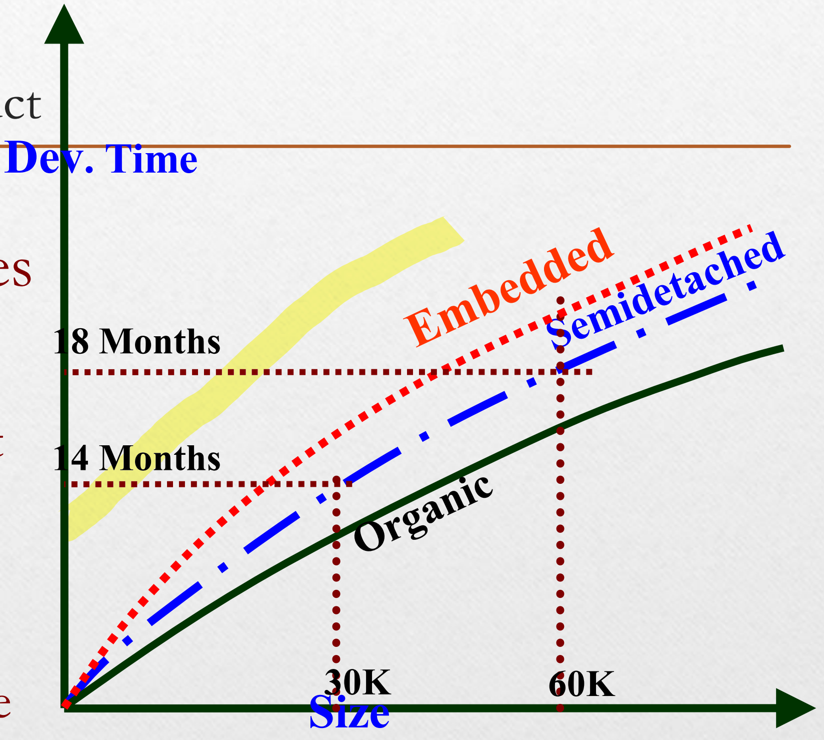
Basic COCOMO Model (CONT.)

- Effort is somewhat super-linear in problem size.



Basic COCOMO Model (CONT.)

- Development time
  - sublinear function of product size.
- When product size increases two times,
  - development time does not double.
- Time taken:
  - almost same for all the three product categories.



Basic COCOMO Model (CONT.)

- Development time does not increase linearly with product size:
  - For larger products more parallel activities can be identified:
    - can be carried out simultaneously by a number of engineers.

Basic COCOMO Model (CONT.)

- Development time is roughly the same for all the three categories of products:
  - For example, a 60 KLOC program can be developed in approximately 18 months
    - regardless of whether it is of organic, semi-detached, or embedded type.
  - There is more scope for parallel activities for system and application programs,
    - than utility programs.

Example

- The size of an organic software product has been estimated to be 32,000 lines of source code.

- Effort = $2.4*(32)^{1.05}$ = 91 PM
- Nominal development time = $2.5*(91)^{0.38}$= 14 months

Intermediate COCOMO

- Basic COCOMO model assumes
  - effort and development time depend on product size alone.
- However, several parameters affect effort and development time:
  - Reliability requirements
  - Availability of CASE tools and modern facilities to the developers
  - Size of data to be handled

Intermediate COCOMO

- For accurate estimation,
  - the effect of all relevant parameters must be considered:
  - Intermediate COCOMO model recognizes this fact:
    - refines the initial estimate obtained by the basic COCOMO by using a set of 15 cost drivers (multipliers).

Intermediate COCOMO (CONT.)

- If modern programming practices are used,
  - initial estimates are scaled downwards.
- If there are stringent reliability requirements on the product :
  - initial estimate is scaled upwards.

Intermediate COCOMO (CONT.)

- Rate different parameters on a scale of one to three:
  - Depending on these ratings,
    - multiply cost driver values with the estimate obtained using the basic COCOMO.

Intermediate COCOMO (CONT.)

- Cost driver classes:
  - Product: Inherent complexity of the product, reliability requirements of the product, etc.
  - Computer: Execution time, storage requirements, etc.
  - Personnel: Experience of personnel, etc.
  - Development Environment: Sophistication of the tools used for software development.

Shortcoming of basic and intermediate COCOMO models

- Both models:
  - consider a software product as a single homogeneous entity:
  - However, most large systems are made up of several smaller sub-systems.
    - Some sub-systems may be considered as organic type, some may be considered embedded, etc.
    - for some the reliability requirements may be high, and so on.

Complete COCOMO

- Cost of each sub-system is estimated separately.
- Costs of the sub-systems are added to obtain total cost.
- Reduces the margin of error in the final estimate.

Complete COCOMO Example

- A Management Information System (MIS) for an organization having offices at several places across the country:
  - Database part (semi-detached)
  - Graphical User Interface (GUI) part (organic)
  - Communication part (embedded)
- Costs of the components are estimated separately:
  - summed up to give the overall cost of the system.

CoCOMO 2

- Provides 3 increasingly detailed cost estimation models.
  - Application composition- prototyping.
  - Early design- higher design stage
  - Post-architecture stage- detailed design and coding stage

Application composition Model

- Effort is estimated as follows,
  - No. of screens, reports, 3GL components of SRS
  - Determine complexity level of each screen complexity based on the following table.

No. of views	Tables<4	Tables<8	Tables>=8
<3	Simple	Simple	Medium
3-7	Simple	Medium	Difficult
>8	Medium	Difficult	difficult

- Determine complexity level of each report complexity based on the following table.

No. of sections	Tables<4	Tables<8	Tables>=8
0 or 1	Simple	Simple	Medium
2 or 3	Simple	Medium	Difficult
4 or more	Medium	Difficult	difficult

- Determine the no. of Object points. (Summation of complexity values for a object)

- Estimate the percentage of reuse expected in a system and compute New Object points. (NOP)
- Determine the productivity rate, Prod = NOP/PM.
- Finally PM is computed as E = NOP/Prod.

- Early design Model
  - Seven cost drivers that characterize the post- architecture model are used.
    - Effort = KSLOC x PI_i $costdriver_i$

- Post architecture model
  - Differs from original COCOMO model by the choice of the cost drivers.
  - Effort = a x $KSLOC^b$ x PI_i $costdriver_i$
  - Where b ranges from 1.01 to 1.26.

Halstead's Software Science

- An analytical technique to estimate:
  - size,
  - development effort,
  - development time.

Halstead's Software Science

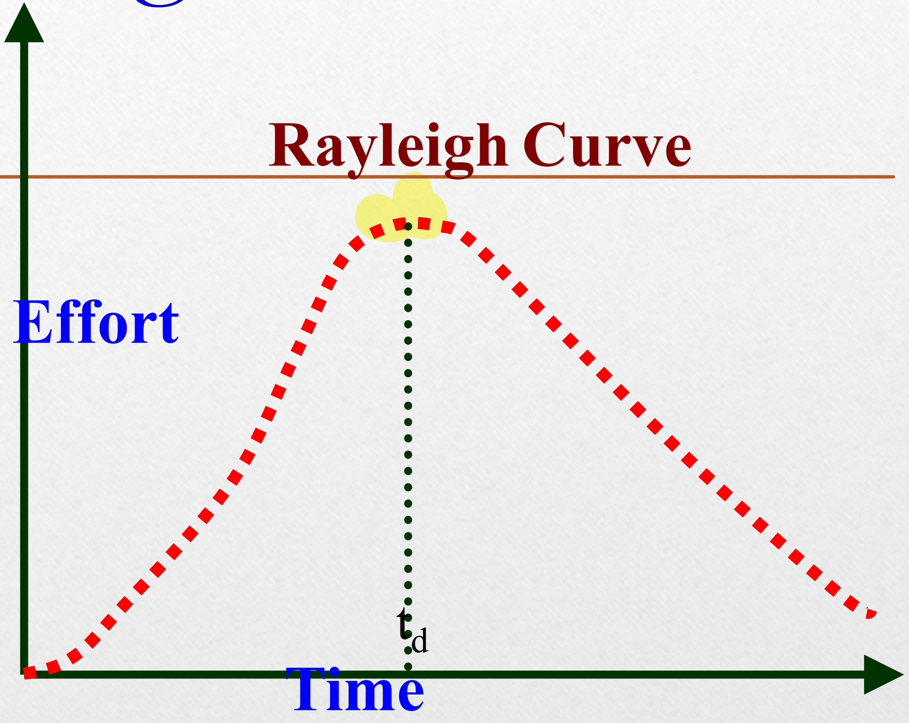
- Halstead used a few primitive program parameters
  - number of operators and operands
- Derived expressions for:
  - over all program length,
  - potential minimum volume
  - actual volume,
  - language level,
  - effort, and
  - development time.

Staffing Level Estimation

- Number of personnel required during any development project:
  - not constant.
- Norden in 1958 analyzed many R&D projects, and observed:
  - Rayleigh curve represents the number of full-time personnel required at any time.

Rayleigh Curve

- Rayleigh curve is specified by two parameters:
  - t_d the time at which the curve reaches its maximum
  - K the total area under the curve.
- $L = f(K, t_d)$



Rayleigh Curve

- Very small number of engineers are needed at the beginning of a project
  - carry out planning and specification.
- As the project progresses:
  - more detailed work is required,
  - number of engineers slowly increases and reaches a peak.

Rayleigh Curve

- From the Rayleigh curve observe that:
  - approximately 40% of the area under the Rayleigh curve is to the left of td and 60% to the right.

Rayleigh Curve

- Putnam observed that:
    - the time at which the Rayleigh curve reaches its maximum value
        - corresponds to system testing and product release.
    - After system testing,
        - the number of project staff falls till product installation and delivery.

Putnam’s Work:

- In 1976, Putnam studied the problem of staffing of software projects:
  - observed that the level of effort required in software development efforts has a similar envelope.
  - found that the Rayleigh-Norden curve
    - relates the number of delivered lines of code to effort and development time.

Putnam's Work (CONT.):

- Putnam analyzed a large number of army projects, and derived the expression:

  $L=C_k K^{1/3} t_d^{4/3}$

  - K is the effort expended and L is the size in KLOC.
  - t_d is the time to develop the software.
  - C_k is the state of technology constant

    reflects factors that affect programmer productivity.

Putnam's Work (CONT.):

- $C_k = 2$ for poor development environment
    - no methodology, poor documentation, and review, etc.
- $C_k = 8$ for good software development environment
    - software engineering principles used
- $C_k = 11$ for an excellent environment

Effect of Schedule Change on Cost

- Using the Putnam's expression for L,

  $K = L^3/(C^3_k \ t^4_d)$

  Or, $K = C/ \ t^4_d$

- For the same product size, $C = L^3/C^3_k$ is a constant.

- Or, $K_1/K_2 = t^4_{d2}/ \ t^4_{d1}$

Effect of Schedule Change on Cost (CONT.)

- Observe:
  - a relatively small compression in delivery schedule
    - can result in substantial penalty on human effort.
- Also, observe:
  - benefits can be gained by using fewer people over a somewhat longer time span.

Example

- If the estimated development time is 1 year, then in order to develop the product in 6 months,
  - the total effort and hence the cost increases 16 times.
  - In other words,
    - the relationship between effort and the chronological delivery time is highly nonlinear.

Effect of Schedule Change on Cost (CONT.)

- Putnam model indicates extreme penalty for schedule compression
  - and extreme reward for expanding the schedule.
- Putnam estimation model works reasonably well for very large systems,
  - but seriously overestimates the effort for medium and small systems.

Effect of Schedule Change on Cost (CONT.)

- Boehm observed:
  - "There is a limit beyond which the schedule of a software project cannot be reduced by buying any more personnel or equipment."
  - This limit occurs roughly at 75% of the nominal time estimate.

Effect of Schedule Change on Cost

(CONT.)

- If a project manager accepts a customer demand to compress the development time by more than 25%
  - very unlikely to succeed.
    - every project has only a limited amount of parallel activities
    - sequential activities cannot be speeded up by hiring any number of additional engineers.
    - many engineers have to sit idle.

Jensen Model

- Jensen model is very similar to Putnam model.
  - attempts to soften the effect of schedule compression on effort
  - makes it applicable to smaller and medium sized projects.

Jensen Model

- Jensen proposed the equation:
  - $L = C_{te} t_d K^{1/2}$
  - Where,
    - C_{te} is the effective technology constant,
    - t_d is the time to develop the software, and
    - K is the effort needed to develop the software.

Project scheduling

- Project scheduling involves deciding which tasks would be taken up when.
- In order to schedule the project activities, a spm needs to do the following,

1. Identify all the tasks needed to complete the project.
2. Break down large tasks into small activities.
3. Determine the dependency among different activities.

4. Establish the most likely estimates for the time durations necessary to complete the activities.

5. Allocate resources to activities.

6. Plan the starting and ending dates for various activities.

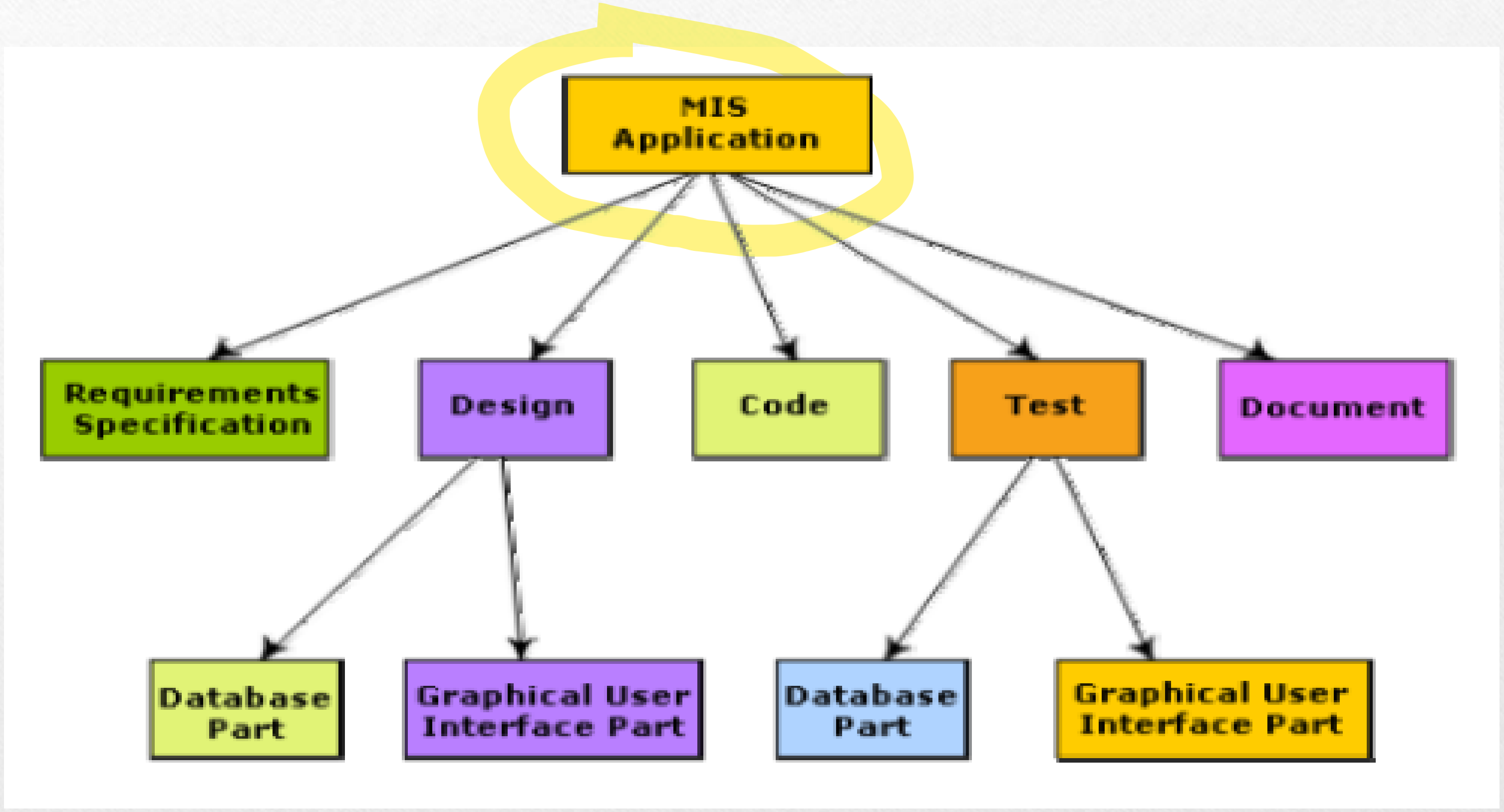
7. Determine the critical path. A critical path is the chain of activities that determines the duration of the project.

Work breakdown structure

- Work Breakdown Structure is used to decompose a given task set recursively into small activities.
- The root of the tree is labelled with the problem name.

IMPORTANT

Work breakdown structure


MIS Application
Requirements Specification
Design
Code
Test
Document
Database Part
Graphical User Interface Part
Database Part
Graphical User Interface Part

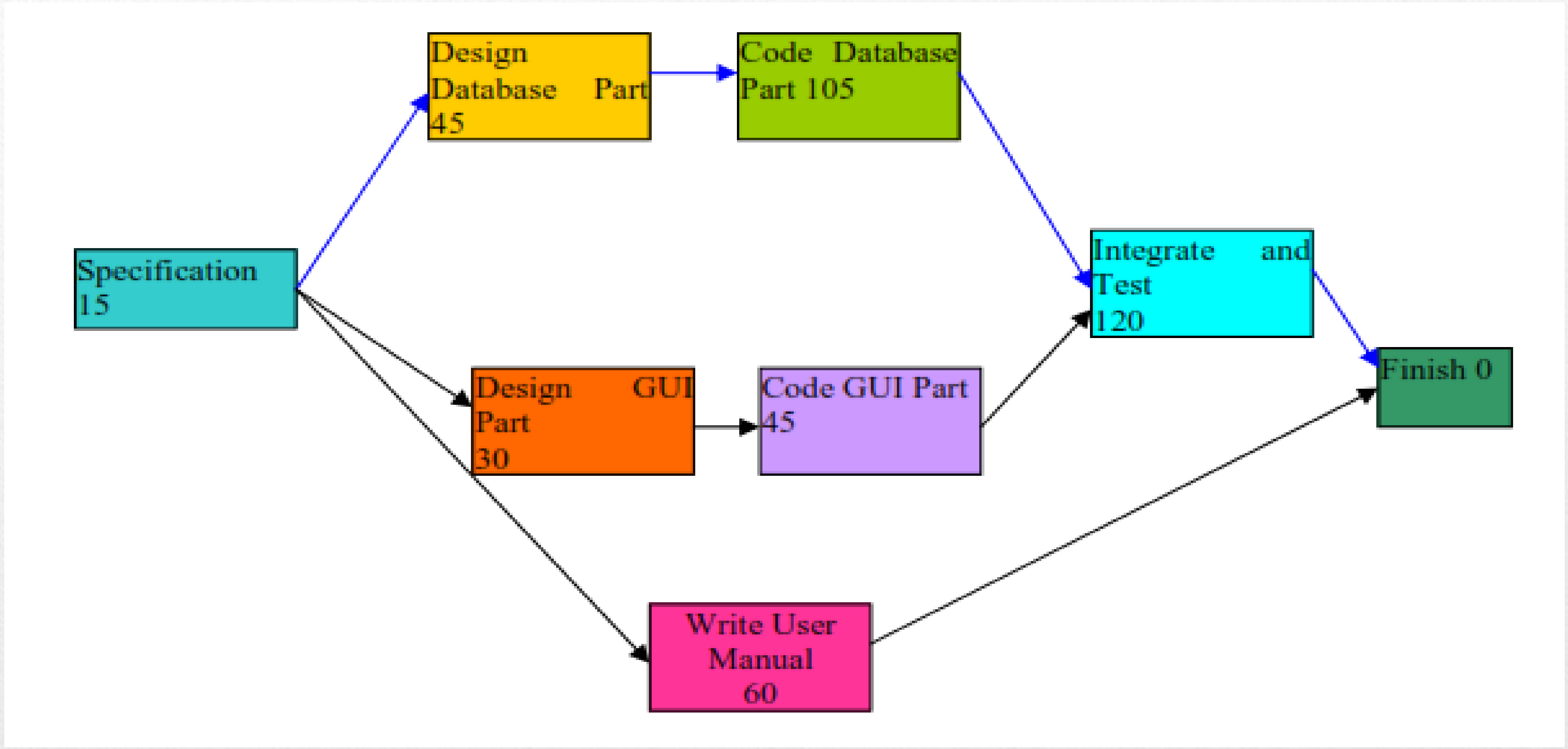
Work breakdown structure of an MIS problem

Activity networks

- An activity network shows the different activities making up the project, their estimated durations and interdependencies.

- Each activity is represented by a rectangular node and the duration of the activity is shown alongside of each task.

Activity networks



Activity network representation of the MIS problem

Critical Path Method (CPM)

This method is used to find the minimal way we can schedule the activities of a project.

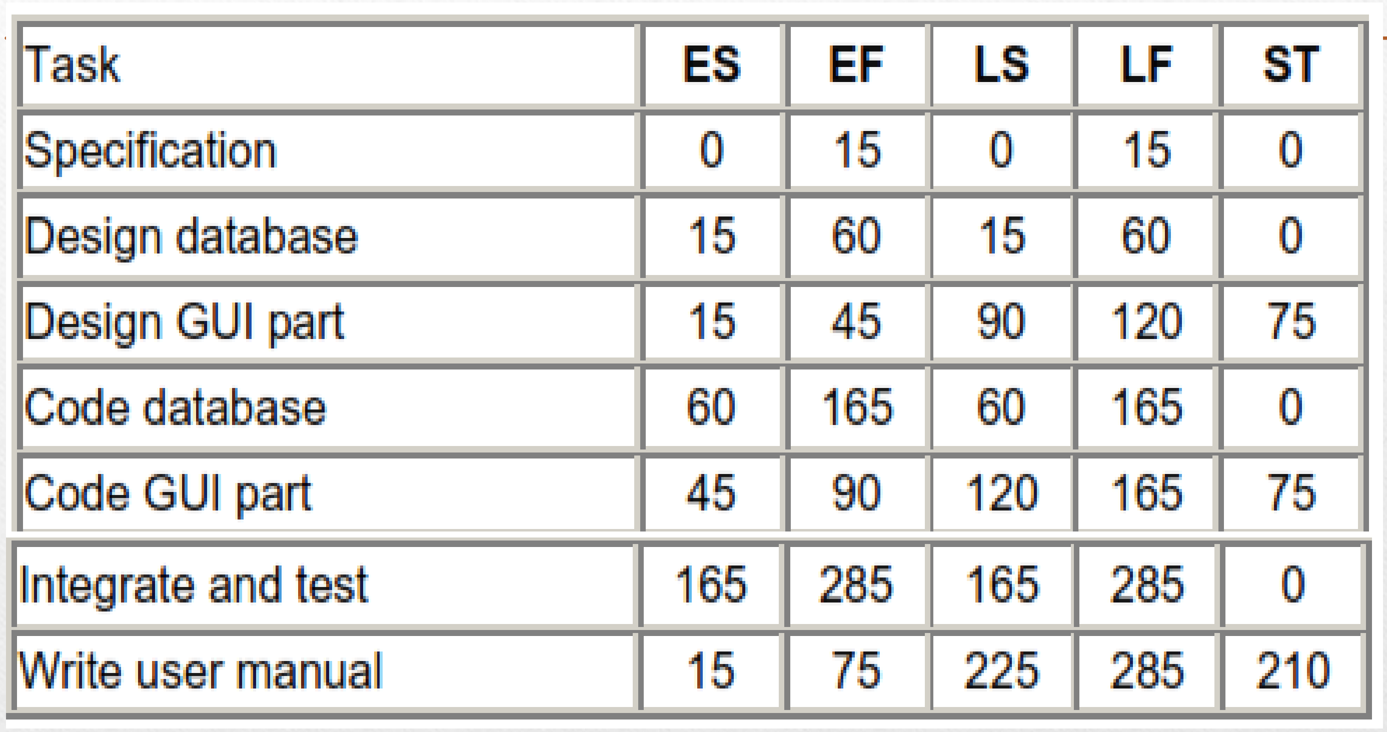
Following terms are used in finding the CPM

- MT(minimum time) – max of all paths from start to finish to complete a project
- ES (earliest start) - max of all paths from start to the task
- LS(latest start)- diff b/w MT and the max of all paths from this task to the finish

Critical Path Method (CPM)

- EF(Earliest Finish)- the sum of the earliest start time of the task and the duration of the task
- LF(Latest Finish)- subtracting max of all paths from this task to finish from MT
- ST(Slack time)- is LS – ES or LF-EF
- Slack time indicates the flexibility in starting and completion of tasks.
- A Critical task is one with a zero slack time.
- A path from the start node to the finish node containing only critical tasks is called a critical path.

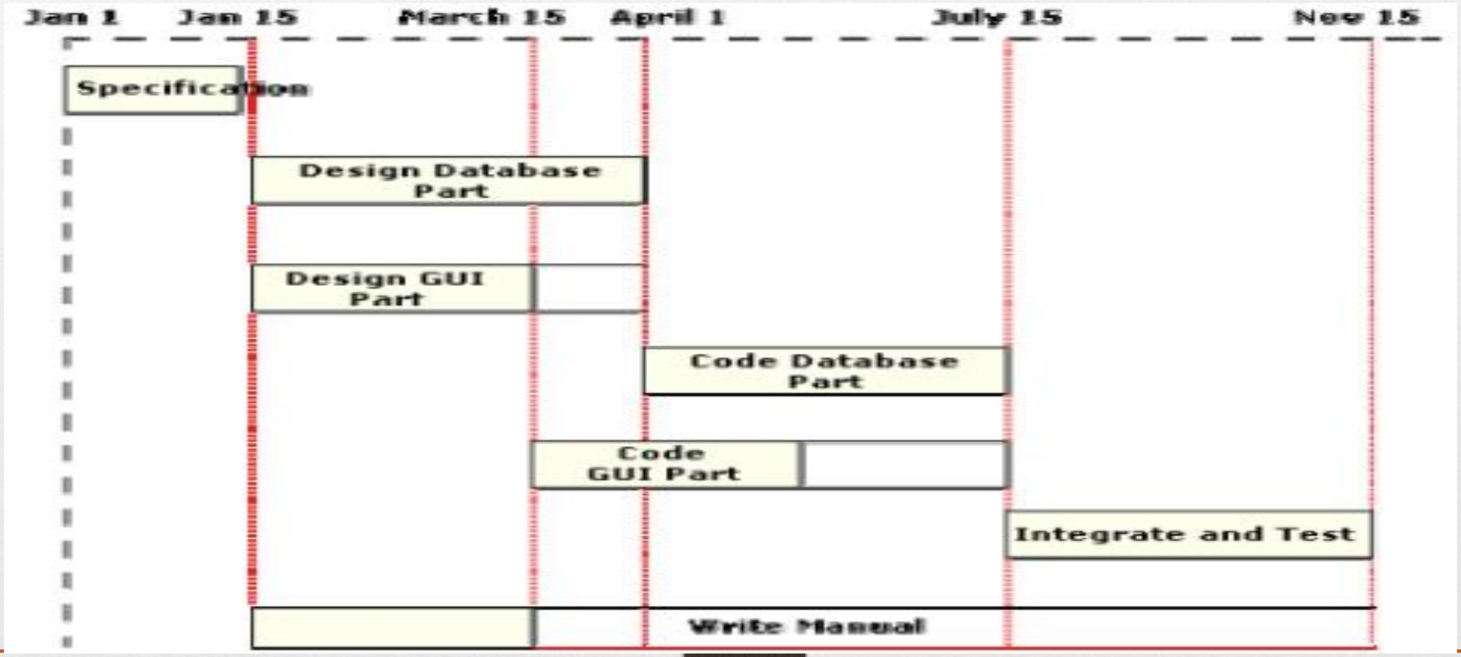
Critical Path Method (CPM)



Task	ES	EF	LS	LF	ST
Specification	0	15	0	15	0
Design database	15	60	15	60	0
Design GUI part	15	45	90	120	75
Code database	60	165	60	165	0
Code GUI part	45	90	120	165	75
Integrate and test	165	285	165	285	0
Write user manual	15	75	225	285	210

Gantt chart

- Gantt charts are mainly used to allocate resources like staff, hardware, software to activities.
- The bars are drawn along a time line.



Gantt chart

- Each bar consists of a white part and a shaded part.
- The shaded part of the bar shows the length of time each task is estimated to take.
- The white part shows the slack time.

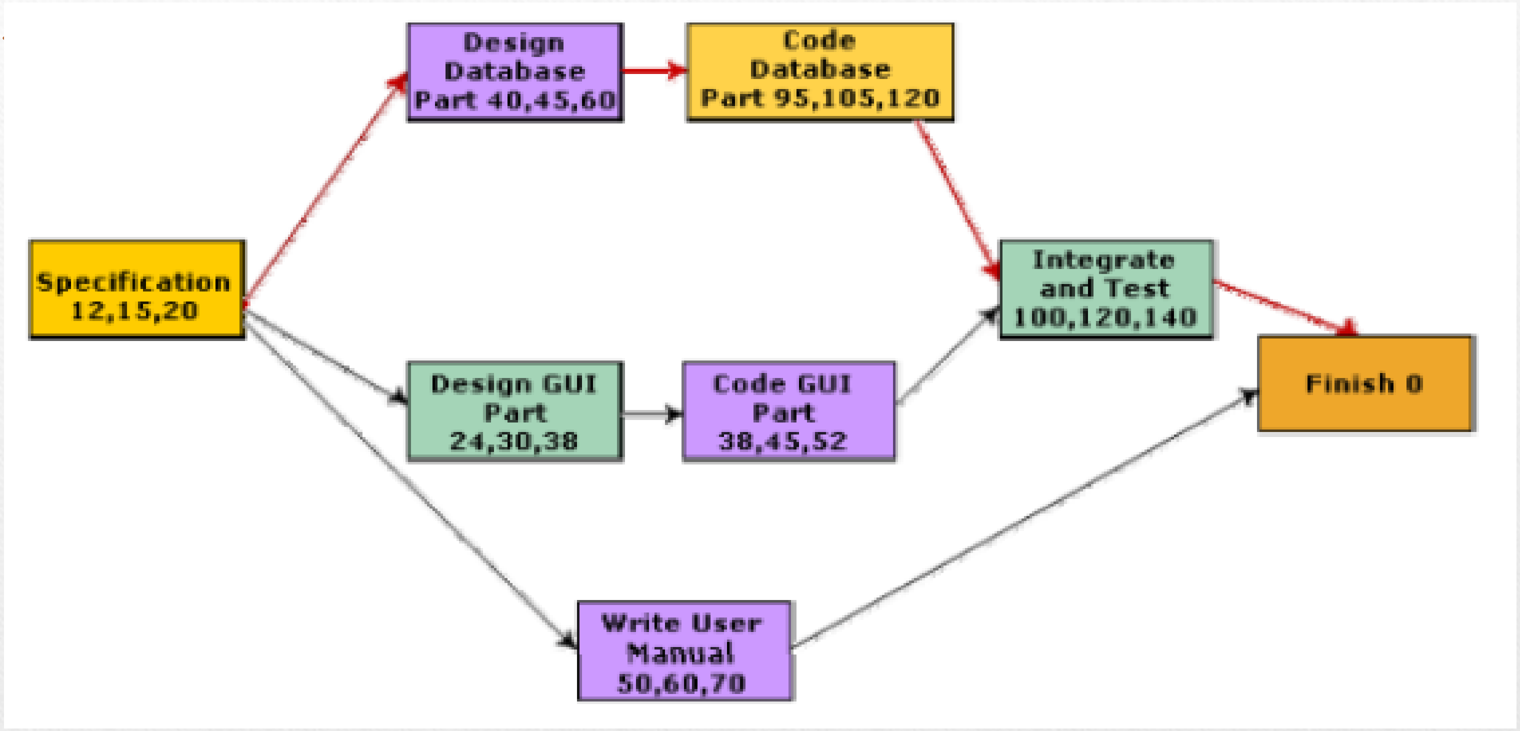
PERT

- PERT (Project Evaluation and Review Technique) charts consist of a network of boxes(activities) and arrows(dependencies).
- The boxes of PERT charts are usually annotated with the

pessimistic, likely, and optimistic estimates for every task.

- PERT charts are a more sophisticated form of activity chart

PERT


Design Database Part 40,45,60
Code Database Part 95,105,120
Specification 12,15,20
Integrate and Test 100,120,140
Finish 0
Design GUI Part 24,30,38
Code GUI Part 38,45,52
Write User Manual 50,60,70

PERT chart representation of the MIS problem

Project Monitoring and Control

- Project manager designates certain key events such as completion of some important activities as Milestones.

- A critical path in this graph is a path along which every milestone is critical to meet the project deadline.

Earned Value Analysis

- A technique for performing quantitative analysis of progress of a project.

- The total hours to do the whole project are estimated and every task is given an earned value based on its estimated percentage of the total.

- Earned value is a measure of progress/percent of completeness.

Earned Value Analysis(EVA)

- EVA is determined with the help of the following steps,
    - Budgeted cost of work scheduled(BCWS) is determined for each work task represented in the schedule.
    - Measured in person-hours or person-days.
    - BCWS(*i*) is the effort planned for work task *i*.
    - BCWS is the sum of all BCWS(*i*) values that should have been completed by that point in time on the project schedule.

Earned Value Analysis(EVA)

- BCWS values for all work tasks are summed to derive the Budget At Completion(BAC).
  - BAC = Σ (BCWS(k)) for all tasks k

- Budgeted cost of work performed (BCWP) is computed as the sum of the BCWS values for all work tasks that have actually been completed by a point in time on the project schedule.

IMPORTANT

Earned Value Analysis(EVA)

- Given the BCWS, BAC and BCWP values, following progress indicators can be computed:
    - Schedule Performance index (SPI) = BCWP/BCWS
    - Schedule variance (SV) = BCWP-BCWS
    - Percent scheduled for completion = BCWS/BAC
    - Percent Complete = BCWP/BAC

Earned Value Analysis(EVA)

- Actual cost of work performed (ACWP) is the sum of effort actually expended on work tasks that have been completed by a point in time on the project schedule.

IMPORTANT

- With the help of this we can compute,
  - Cost performance index CPI = BCWP/ACWP
  - Cost variance CV = BCWP - ACWP

Earned Value Analysis(EVA)

- An SPI value close to 1.0 indicates efficient execution of the project schedule.

- Illy, A CPI value close to 1.0 provides a strong indication that the project is within its defined budget.

- EVA helps project manager to identify scheduling difficulties before they become apparent.

Organization Structure

- Functional Organization:
  - Engineers are organized into functional groups, e.g.
    - specification, design, coding, testing, maintenance, etc.
  - Engineers from functional groups get assigned to different projects

Advantages of Functional Organization

- Specialization
- Ease of staffing
- Good documentation is produced
  - different phases are carried out by different teams of engineers.
- Helps identify errors earlier.

Project Organization

- Engineers get assigned to a project for the entire duration of the project
  - Same set of engineers carry out all the phases
- Advantages:
  - Engineers save time on learning details of every project.
  - Leads to job rotation

Team Structure

- Problems of different complexities and sizes require different team structures:
  - Chief-programmer team
  - Democratic team
  - Mixed organization

Democratic Teams

- Suitable for:
  - small projects requiring less than five or six engineers
  - research-oriented projects
- A manager provides administrative leadership:
  - at different times different members of the group provide technical leadership.

Democratic Teams

- Democratic organization provides
  - higher morale and job satisfaction to the engineers
  - therefore leads to less employee turnover.

- Suitable for less understood problems,
  - a group of engineers can invent better solutions than a single individual.

Democratic Teams

- Disadvantage:
  - team members may waste a lot time arguing about trivial points:
    - absence of any authority in the team.

Chief Programmer Team

- A senior engineer provides technical leadership:
  - partitions the task among the team members.
  - verifies and integrates the products developed by the members.

Chief Programmer Team

- Works well when
  - the task is well understood
    - also within the intellectual grasp of a single individual,
  - importance of early completion outweighs other factors
    - team morale, personal development, etc.

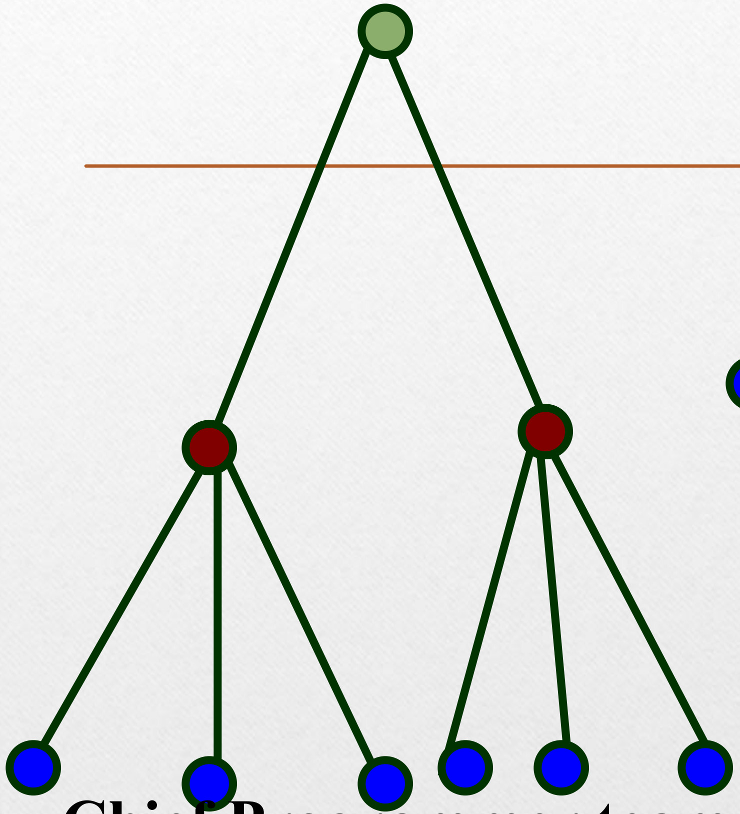
Chief Programmer Team

- Chief programmer team is subject to single point failure:
  - too much responsibility and authority is assigned to the chief programmer.

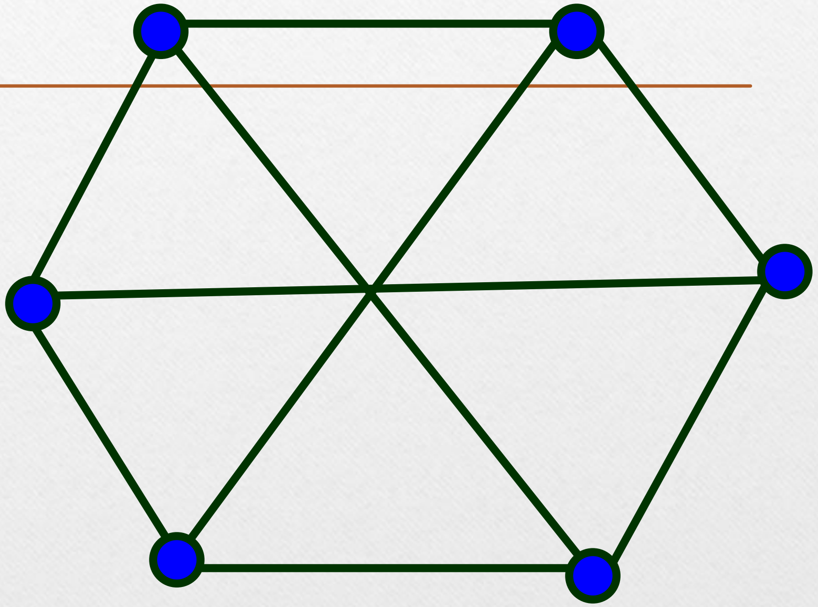
Mixed Control Team Organization

- Draws upon ideas from both:
  - democratic organization and
  - chief-programmer team organization.
- Communication is limited
  - to a small group that is most likely to benefit from it.
- Suitable for large organizations.

Team Organization

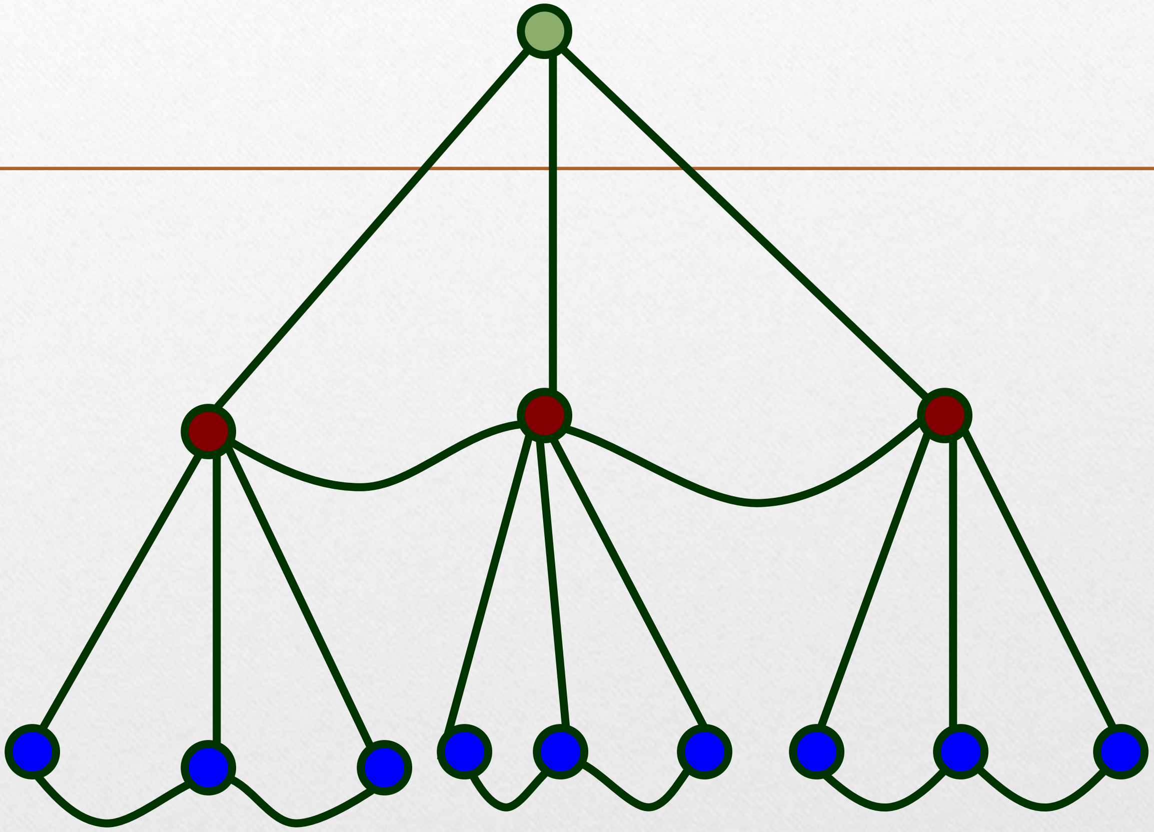


Chief Programmer team



Democratic Team

Mixed team organization



Qualities of a Software Developer

- Familiarity with SE principles
- Good technical knowledge of the project areas
- Good Programming abilities
- Good communication skills like oral, written, interpersonal skills
- High motivation
- Sound knowledge of fundamentals of CS
- Intelligence, Ability to work in a team, Discipline.

Risk Management

- A risk is any anticipated unfavourable event or circumstance that can occur while a project is underway.

- Risk Management is done in three ways
  - Risk identification
  - Risk Assessment
  - Risk Containment

Risk Management

- Risk Identification
  - Project risks
  - Technical risks
  - Business risks

IMPORTANT

- Risk Assessment
  - Prioritise the risks
    - $P = r * s$
    - Where, r is likelihood of risk and s is consequence of risk

Risk Management

- Risk Containment
  - Avoid the risk
    - Discuss with customer to change requirements
  - Transfer the risk
    - Component developed by third party, buying insurance cover.

IMPORTANT

  - Risk reduction
    - Is done with the help of risk leverage given as

$$= \frac{\text{risk exposure before reduction} - \text{risk exposure after reduction}}{\text{Cost of reduction}}$$

Software Configuration Management

- In software development there would be numerous intermediate outputs, checking the state of these outputs at any point of time is called Software Configuration.

- Software configuration management deals with effectively tracking and controlling the configuration of a software product during its life cycle.

Software Configuration Management

- SCM addresses the following problems
  - Inconsistency problem
  - Problems associated with concurrent access
  - Providing a stable development environment
  - System accounting and maintaining status information

Summary

- We discussed the broad responsibilities of the project manager:
  - Project planning
  - Project Monitoring and Control

Summary

- To estimate software cost:
  - Determine size of the product.
  - Using size estimate,
    - determine effort needed.
  - From the effort estimate,
    - determine project duration, and cost.

Summary (CONT.)

- Cost estimation techniques:
  - Empirical Techniques
  - Heuristic Techniques
  - Analytical Techniques
- Empirical techniques:
  - based on systematic guesses by experts.
    - Expert Judgement
    - Delphi Estimation

Summary (CONT.)

- Heuristic techniques:
  - assume that characteristics of a software product can be modeled by a mathematical expression.
  - COCOMO
- Analytical techniques:
  - derive the estimates starting with some basic assumptions:
  - Halstead's Software Science

Summary (CONT.)

- The staffing level during the life cycle of a software product development:
  - follows Rayleigh curve
  - maximum number of engineers required during testing.

Summary (CONT.)

- Relationship between schedule change and effort:
  - highly nonlinear.
- Software organizations are usually organized in:
  - functional format
  - project format

Summary (CONT.)

- Project teams can be organized in following ways:
  - Chief programmer: suitable for routine work.
  - Democratic: Small teams doing R&D type work
  - Mixed: Large projects