CSE1005: Software Engineering

Module No. 5: Managing Software Projects

Dr. K Srinivasa Reddy
SCOPE, VIT-AP University

Module No. 5: Managing Software Projects

People, Product, Project, Process(4P), Software Project Estimation, Decomposition Technique, Empirical Estimation Models, Project Scheduling.

Text Book:

1. Roger Pressman, “Software Engineering: A Practitioner’s Approach”, McGraw-Hill, 7th Edition, 2016.

Course Outcome: Understand and apply the project management techniques (CO5)

Managing Software Projects – Management Spectrum

Example:

- **Going to the tour with friends**

What is it?

- Project management involves the **planning, monitoring, and control of the people, process, and events** that occur as software evolves from a **preliminary concept to full operational deployment.**
- **Primary Objectives:**
- To meet specified performance
  - ... within cost
  - ... and on schedule

Managing Software Projects

- **Who does it?**
  - **Software engineer:** Manages day-to-day activities, planning, monitoring, and controlling technical tasks.
  - **Project managers:** Plan, monitor, and control the **work of a team of software engineers.**
  - **Senior managers:** Coordinate the interface between the business and software professionals.

- **Why is it important?**
  - Building computer software is a complex task, which involves many people working over a long time. Hence, software projects need to be managed.

Managing Software Projects – Management Spectrum

- Describes the **management of a software project**.
  - The management of a software project **starts** from requirement analysis and **finishes** based on the nature of the product, it **may or may not end** because almost all software products **faces changes and requires support.**
  - It is about **turning the project from plan to reality**.
- Focuses on the **four P's:**
  - People
  - Product
  - Process
  - Project
- The **manager of the project** has to **control all P's to have a smooth flow** in the **progress of the project** and **to reach the goal**.

Managing Software Projects – Management Spectrum

- **People:** The **most important element** of a successful project.

- **Product:** The software **to be built**.
  - **Context -** How it fits into larger system
  - **Information Objectives -** What data comes into and goes out of the software
  - **Function and Performance**

- **Process:** The **set of framework activities (communication, planning, modeling, construction, deployment)** and **software engineering tasks** which are **appropriate for people and product** to get the job done (Paradigm used, tasks, work products, mile stones etc.)

- **Project: All work** required **to make product a reality**. (Planning, Monitoring, Controlling, etc.)

Managing Software Projects – Management Spectrum

- **What among these 4Ps is the most important contributor to a successful software project? (**In 1988, the engineering vice presidents of three major technology companies were asked about the most important contributor to a successful software project)
  - **VP1:** It's not the tools we use, but it's the **people.**
  - **VP2:** It is having smart **people**. Very little else matters in my opinion. The most important thing you do for a Project is **selecting the staff**.
  - **VP3:** The only rule I have in management is to ensure I have really good **people**. Then, I provide an environment in which good people can work.

Managing Software Projects – Management Spectrum - People

- **People:**
  - The **primary resources** of the projects.
  - Includes from **manager to developer, from customer to end user**.
  - **People Capability Maturity Model (People-CMM)** defines
    - Staffing,
    - Communication and coordination,
    - Work environment,
    - Performance management,
    - Training,
    - Compensation,
    - Competency analysis and development,
    - Career development,
    - Workgroup development,
    - Team / Culture development, and others.

Organizations that achieve **high levels of People-CMM maturity** have a **higher possibility** of implementing effective software project management practices.

Managing Software Projects – Management Spectrum

- **People:**
  - **First step to a successful project:** Identifying the **roles of people** to play in a project.
  - **Stakeholders : The software process (and every software project) is populated by stakeholders who can be categorized as follows:**
    - **Senior Managers :** Define the business issues which influence the project.
    - **Project (Technical) managers :** Plan, motivate, organize and control the practitioners who do software work.
    - **Practitioners :** Deliver the technical skills that are necessary to engineer a product or application.
    - **Customers:** Specify the requirements for the software to be engineered and other stakeholders who have a peripheral interest in the outcome.
    - **End-users:** Interact with the final product once it is released for production use.

Team Leaders

Managing Software Projects – Management Spectrum - People:

VIT-AP UNIVERSITY

Team Leaders :

- Leadership is an integral part of project management.
- Should feel a **sense of ownership** on the project and **take to heart the aspect of leading a team.**
- **MOI** model of leadership:
  - **Motivation: Encourage** technical people to produce to their best ability.
  - **Organization**: **Ability to mold existing processes** (or invent new ones) that will **enable the initial concept to be translated into a final product**.
  - **Ideas or Innovation: Ability to encourage people to create and feel creative** even when they must work within bounds established for a particular software product or application.

Managing Software Projects – Management Spectrum -People:

An effective project manager emphasizes four key traits:

Problem solving:

- Diagnose the technical and organizational issues.
- Skills to structure a solution
- Motivate other practitioners to develop the solution,
- Apply lessons learned from past projects to new situations
- Remain flexible enough to change direction if initial attempts at problem solution are fruitless.

Managerial identity:

- A good project manager must take charge of the project.
- Must have the confidence to control when necessary
- Assurance to allow good technical people to follow their instincts.

Achievement:

- Must reward initiative and accomplishment to optimize the productivity of a project team.
- Must demonstrate through own actions that controlled risk taking will not be punished.

Influence and team building:

- An effective project manager must be able to “read” people
- Must be able to **understand verbal and nonverbal signals** and react to the needs of the people sending these signals.
- The manager must **remain under control in high-stress** situation.

Managing Software Projects – Management Spectrum -People:

VIT-AP UNIVERSITY

The Software Team

To achieve a high-performance team:

- Team members must have trust in one another
- The distribution of skills must be appropriate to the problem
- Mavericks may have to be excluded from the team, if team cohesiveness is to be maintained.
- However, **unfortunately**, many teams suffer from **"team toxicity"**

Managing Software Projects – Management Spectrum

Five factors that "foster a potentially toxic team environment":

1. A **frenzied work atmosphere**: Team members waste energy and lose focus on the objectives of the work to be performed.
2. High **frustration** (caused by personal, business, or technological factors) that causes friction among team members,
3. A **fragmented or poorly coordinated** software process, or a poorly defined or improperly chosen process model that becomes a roadblock to accomplishment.
4. An **unclear definition** of roles on the software team, (resulting in a lack of accountability and resultant finger-pointing) and
5. **Continuous and repeated exposure** to failure, leads to a loss of confidence and a lowering of morale.

Q: How these can be avoided / overcome?

Recap

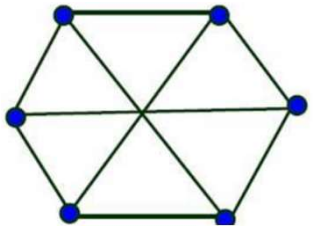
Module – 5: Managing Software Projects

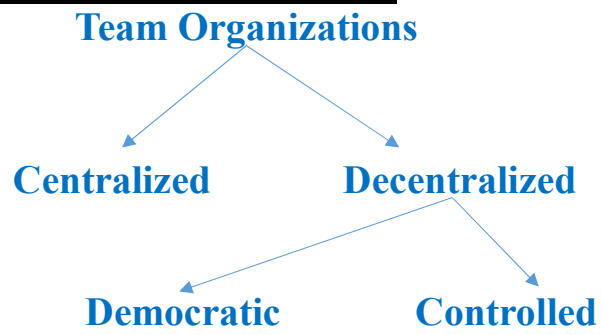
- Management Spectrum - People, Product, Project, Process(4P's)
- People – Stakeholders
  - Team Leaders
  - Important Characteristics
    - Problem solving
    - Managerial identity
    - Achievement
    - Influence and team building
  - The Software Team
    - Toxic team environment and Solution
  - Team Organizations and Organization Paradigms

“Team Organizations” for software engineering teams:

Democratic Decentralized (DD):

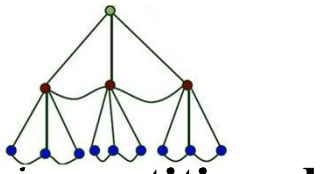
- **No permanent leader**
- **Task coordinators** appointed for **short duration and then replaced by others**
- **Decisions made by** group consensus
- Horizontal communication **(no hierarchy)**
- Better for **difficult problems**

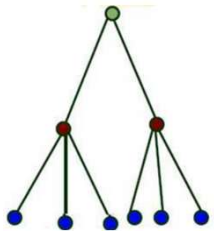




Controlled Decentralized (CD):

- **Defined leader** to coordinates tasks and
- **Secondary leaders** responsible for **subtasks**
- **Problem solving - Implementation** of solution is **partitioned among subgroups** by the Team Leader – Group Activity
- **Horizontal communication** among **subgroups**
- **Vertical communication** along **control hierarchy**





Controlled Centralized (CC):

- The **defined leader** manages **top level problem solving and internal team coordination**
- **Vertical communication** between leader and team members.

“Team Organizations” for software engineering teams:

- **Seven Factors to be considered when selecting a software project team structure**
  - The **difficulty** of the problem to be solved
  - The **size** of the resultant program(s) in lines of code or function points
  - The **duration** that the team will stay together (team lifetime)
  - The **degree** to which the problem can be **modularized**
  - The **required quality** and **reliability** of the system to be built
  - The **inflexibility** of the **delivery date**
  - The degree of **sociability (communication)** required for the project

Group Structure	Difficulty		Size		Duration		Modularity		Reliability		Time		Sociability	
	High	Low	Large	Small	Short	Long	High	Low	High	Low	Strict	Lax	High	Low
Democratic Decentralized	X			X		X		X	X			X	X	
Controlled Decentralized		X	X		X		X		X			X		X
Controlled Centralized		X	X		X		X			X	X			X

Seven Factors

"Organizational paradigms" for software engineering teams:

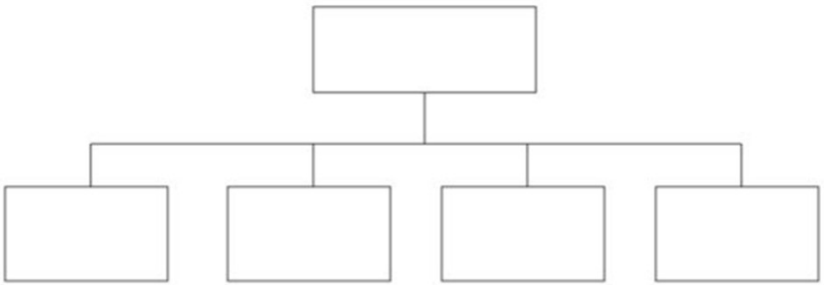
VIT-AP UNIVERSITY

Closed paradigm (Similar to a Controlled Centralized):

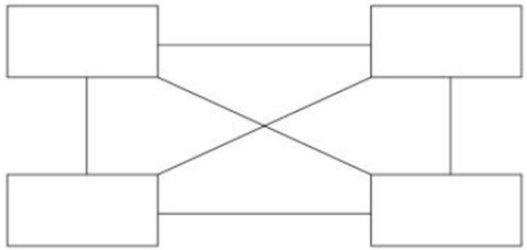
- Structures a team along a **traditional hierarchy of authority**
- Teams can **work well** when **producing software similar to past efforts**
- **Less likely to be innovative**
- Works easily with **orderly performance (organizational chart – top down decisions)**
- Organizations are oriented toward **stability, continuity and security.**

Random paradigm:

- Structures a team **loosely** and **depends** on **individual initiative** of the team members
- Teams with **Innovation or technological breakthrough will excel**
- Struggle when "orderly performance" is required
- **Less stable**



Hierarchical



Democratic Organization - **random paradigm**

"Organizational paradigms" for software engineering teams:

Open paradigm:

- Structures a team to achieve **controls (closed paradigm)** and **innovation (random paradigm)**
- **Integrates stability with innovation** and **individual with collective interests**
- **Collaborative work is performed** with heavy communication and **consensus-based decision-making**
- Time pressure and familiarity with the work is good
- Well suited to the solution of **complex problems**

Synchronous paradigm:

- **Relies** on the natural categorization of a problem i.e., **Suitable for problems that can be divided easily into sub problems.**
- Features **harmonious alignment of individuals with a shared vision of goals and methods**
- **Organizes** team members to **work on pieces of the problem with less communication among themselves.**

You have been appointed a project manager for a small software products company. Your job is to build a breakthrough product that combines virtual reality hardware with state-of-the-art software. Because competition for the home entertainment market is intense, there is significant pressure to get the job done. What team structure would you choose and why? What software process model(s) would you choose and why?

Key Terms: Project Manager, Small software, Competition for the home entertainment market is intense, significant pressure to get the job done.

Answer:

- Choose **Controlled Decentralized (CD) / Closed Paradigm:** A defined leader who co-ordinates specific tasks.
- Producing software that is **quite similar to past efforts** and less likely to be innovative.
- Process Model: **The spiral model**.

You have been appointed a project manager for a major software products company. Your job is to manage the development of the next-generation version of its widely used word-processing software. Because competition is intense, tight deadlines have been established and announced. What team structure would you choose and why? What software process model(s) would you choose and why?

Key Terms: Project Manager, major software, development of Next-Generation, Competition is intense, tight deadlines.

Answer:

- An **Open paradigm** team structure
- Open paradigm - Structures a team that achieves some of the controls but also much of the innovation. Here time pressure and familiarity with the work is good.
- An **incremental process model** is indicated by the **deadline driven nature of this work.**

You have been appointed a software project manager for a company that services the genetic engineering world. Your job is to manage the development of a new software product that will accelerate the pace of gene typing. The work is R&D oriented, but the goal is to produce a product within the next year. What team structure would you choose and why? What software process model(s) would you choose and why?

Key Terms: Project Manager, development of a **new software** product that will accelerate the pace of gene typing, R&D oriented, timelines

Answer:

- Preferred **Random team structure** because here work is innovative type
- Another possibility is to use an **Open paradigm** team structure.
- An **incremental process model or an evolutionary process model** could be used, given the deadline driven nature of this work.

4P's - Product:

- **Product:** The software **to be built**.
  - **Context -** How it fits into larger system i.e., The **functions and features that are to be delivered to end users**
  - **Information Objectives -** The **data** that are input to and output from the system.
  - **Function -** The **content** that is presented to users as a **consequence of using the software.**
  - The **performance, constraints, interfaces, and reliability** that bound the system.
- Must be examined at the very beginning of the software project.
- **Two major activities**
  - Determination of software scope and
  - Problem Decomposition

Product - Determination of Software Scope :

- Describes **unambiguous and understandable** at the management and technical levels.
- Defined using **two techniques**
  - A **narrative description** is developed after communication with all stakeholders
  - A **set of use cases** is developed by end users
- Statement must be **bounded i.e.,**
  - **Quantitative data** (e.g., number of simultaneous users, target environment, maximum allowable response time) are stated explicitly
  - **Constraints and / or limitations** (e.g., product cost restricts memory size) are noted
  - **Mitigating factors** (e.g., desired algorithms) are described.

Product - Problem Decomposition (Partitioning or Problem elaboration)

- Sits at the **core** of software **requirements analysis**
- Depends on **project complexity**
- Involves **outlining of work tasks** involved in **each process framework activity**
- Applied in **two major areas**
  - **Functionality** that must be delivered
  - The **process** used to **deliver** it

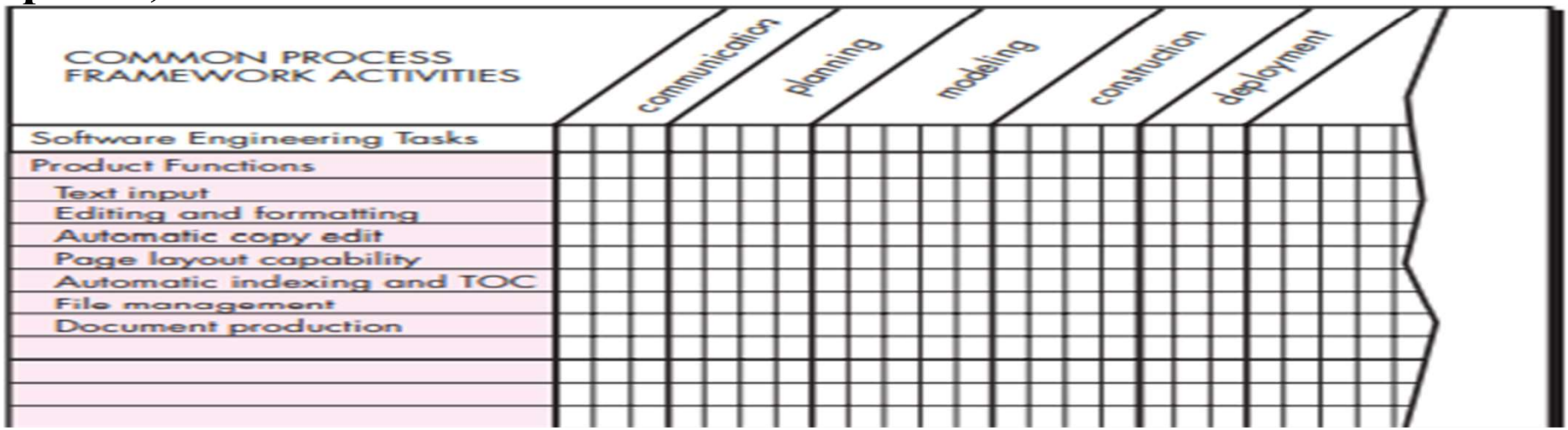
- **To develop a reasonable project plan**, decompose the problem into
  - List of functions / use-cases
  - User stories

4P's - Process

- The **set of framework activities (communication, planning, modeling, construction, deployment)** and **software engineering tasks** which are **appropriate for people and product** to get the job done.
- The **project manager** must decide **which process model** is most appropriate for the **people and the product** should be selected.
  - Different **process models**: Linear sequential, Prototyping, RAD, Spiral, …
  - Project manager must **decide which model to use** depending on
    - Customers who have requested the product
    - People who would work on project
    - Product characteristics
    - Project development environment

Process : Melding the Product and the Process

- Project **planning begins with melding the product and the process**
- **Each function** to be engineered **must pass through the set of framework activities** defined for a software organization **listed in the top row.**
- Each **major product function** is listed in the **left-hand column**.
- Software engineering **work tasks** would be **entered in the following row.**
- The **job of the project manager** is to **estimate** resource requirements **for each matrix cell, start and end dates for the tasks associated with each cell, and work products to be produced as a consequence of each task, quality assurance points, and milestones.**



4P's - Project:

- **Project: All work** required **to make product a reality**. (Planning, Monitoring, Controlling, etc.)

Projects get into trouble or signs that a project is in jeopardy:

- Software people don't understand their customer's needs.
- The product scope is poorly defined.
- Changes are managed poorly.
- The chosen technology changes.
- Business needs change [or are ill-defined].
- Deadlines are unrealistic.
- Users are resistant.
- Sponsorship is lost [or was never properly obtained].
- The project team lacks people with appropriate skills.
- Managers [and practitioners] avoid best practices and lessons learned

Project : Commonsense approach to manage software projects

VIT-AP UNIVERSITY

How to avoid Problems?

1. **Start on the right foot**
   - Involves detailed understanding of project
   - Setting realistic objectives & expectations
   - Selecting the right team
   - Facilitating the team
2. **Maintain Momentum**
   - Provide incentives
   - Reduce bureaucracy and give autonomy to team members but with supervision
3. **Track Progress**
   - Progress is tracked as work products are produced and approved as part of a Quality Assurance activity.
4. **Make smart decisions**
   - Use existing software components, choose standard approaches and **keep it simple**
   - Avoid risks and allocate more time than needed for complex tasks
5. **Conduct a postmortem analysis**
   - Establish a **consistent mechanism for extracting lessons learned for each project.**
   - **Compare** planned and actual schedule
   - Collect and analyze project metrics
   - Get feedback from team and customers

Software Project Estimation

What is it?

- A real need for software has been established; stakeholders are onboard, software engineers are ready to start, and the project is about to begin.
- **But how do you proceed?**
- Software project planning encompasses five major activities - **estimation, scheduling, risk analysis, quality management planning, and change management planning**.
- **Estimation:** An attempt **to determine how much** money, effort, resources, and time it will take to build a specific software-based system or product.

Why is it important?

- House can't be built without knowing
  - How much to spend?
  - The tasks need to perform, and
  - The time line for the work to be conducted?
  - **Hence, it is reasonable to develop an estimate before creating the software**

Software Project Estimation

What are the steps?

- Estimation begins with
  - **Description of the scope of the problem.**
  - **Decomposition of the problem into a set of smaller problems**, and each of these is estimated using historical data and experience as guides.
- **Problem complexity and risk** are considered before a final estimate is made.

The **accuracy** of a **software project estimate depends on**:

- The **degree** to which the planner has properly **estimated the size (KLOC)** of the product to be built
- The **ability to translate the size estimate** into **human effort, calendar time, and money**
- The **degree** to which the **project plan** reflects the **abilities of the software team**
- The **stability** of both product **requirements** and the **environment** that supports the software engineering effort.

Software Project Estimation

To achieve reliable cost and effort estimates, a number of options arise:

1. **Delay estimation** until late in the project (we should be able to achieve 100% accurate estimates after the project is complete)
2. Base estimates on **similar projects** that have already been completed.
3. Use simple **decomposition techniques** for project **cost** and **effort** estimates.
4. Use **empirical estimation models** for software **cost** and **effort** estimation.

Option #

1. Not Practical, but results in good numbers
2. Work reasonably well, but **past experience not always been a good indicator of future results.**

3 & 4. Viable approaches to software project estimation.

- Decomposition techniques take divide-and-conquer approach for software project estimation.
- **Cost and Effort** estimation can be done in a **stepwise fashion** by decomposing a project into major functions, sub functions and related software engineering activities,
- **Empirical estimation models complement** (**helping**) decomposition techniques and offer a potentially valuable estimation approach **if the historical data used to seed the estimate is good.**

Software Project Estimation

VIT-AP UNIVERSITY

A model is based on experience (historical data) and takes the form

$$\mathbf{d = f(v_i)}$$

where

d = One of a number of estimated values (e.g., effort, cost, project duration)

v_i = Selected independent parameters (e.g., estimated LOC, FP, use case, process)

Recap

Module – 5: Managing Software Projects

- Management Spectrum (4P's)
- People
  - Team Leaders & Important Characteristics
  - The Software Team - Toxic team environment and Solution
  - Team Organizations and Organization Paradigms
- Product
  - Determination of software scope and
  - Problem Decomposition
- Process - Melding the product and the process
- Project - Reasons for Project Fail and Solution
- **Software Project Estimation**
  - **Decomposition techniques**
  - **Empirical estimation models**

Software Project Decomposing

VIT-AP UNIVERSITY

- **Software project estimation** is a form of **problem solving**
- Hence **decompose the problem -** Recharacterizing it as a **set of smaller problems** is required.
- Before an estimate can be made, the **project planner**
  - Must **understand the scope** of the software to be built
  - **Generate an estimate** of the software size.
- **Decomposition Techniques:**
  - Software **sizing**
  - **Problem** based estimation
    - Based on either source **Lines of Code (LOC), Function Point (FP)** estimates
  - **Process** based estimation
    - Based on **effort required** to accomplish the task
  - Estimation with **use cases**

Software Project Decomposing - Decomposition Techniques – Sizing

- **Function point sizing (Indirect approach)**
  - Develop estimates of the information domain characteristics
- **Standard component sizing (Direct approach)**
  - **Estimate the number of occurrences of** each standard component
  - **Use historical project data** to determine the **delivered LOC size per standard component**
    - Ex. E-Commerce application, payment gateway – no need to develop, merge
- **Change sizing**
  - **Used when changes** are being made to existing software
  - **Estimate** the **number and type of modifications** that must be accomplished
  - Types of modifications: reuse, adding, changing and deleting code
  - An **effort ratio used to estimate** each **type of change and the size of the change**

- The **results** of the estimates are used to compute an **optimistic (low)**, **a most likely,** and a **pessimistic (high)** value for **software size.**

Software Project Decomposing - Decomposition Techniques

VIT-AP UNIVERSITY

Problem-Based Estimation

- Every project will have scope,
  - Statements written as part of scope are bounded [Statements shouldn't be like open ended statements]
  - Example: **Hey, Tomorrow I am going to meet that friend.**
    - Which friend? Boy / Girl
    - Name of the Friend and details are required.
- **Steps:**
  - Start with a **bounded statement** of scope
  - Decompose the software into **problem functions** that can be estimated individually
  - Compute **LOC or FP** value for each function.
  - Derive **cost & effort** estimates by applying the **LOC or FP values** to baseline productivity metrics. (Example: LOC / Person – Month, FP / Person – Month)
  - **Combine** function estimates to produce an overall estimate of the entire project.

Software Project Decomposing - Decomposition Techniques

Problem-Based Estimation

- In general, the **LOC/pm and FP/pm** metrics should be computed by **project domain**
  - Important factors are **team size, application area, and complexity**
- FP and LOC estimation differ in the **level of detail** required for decomposition with each value
  - For **FP, decomposition occurs** for the **five information domain characteristics (**EI, EO, EQ, ILF, EIF) and the **14 adjustment factors**
  - For **LOC**, **decomposition of functions** is essential and **should go into considerable detail (the more detail, the more accurate the estimate)**
- For both approaches,
  - The **planner uses lessons learned to estimate** an **optimistic, most likely, and pessimistic** size value for each function or count
  - The **three-point or expected size value S** = $(S_{opt} + 4S_m + S_{pess}) / 6$
  - Historical LOC or FP data is then compared to S in order to cross-check it.

Software Project Decomposing - Decomposition Techniques

Problem-Based Estimation : LOC-Based Estimation

VIT-AP UNIVERSITY

Example:
A software package to be developed for a computer-aided design application for mechanical components. The software is to execute on a desktop workstation and must interface with various computer graphics peripherals including a mouse, digitizer, high-resolution color display, and laser printer.

Example of LOC-Based Estimation

Example:

A software package to be developed for a computer-aided design application for mechanical components. The software is to execute on a desktop workstation and must interface with various computer graphics peripherals including a mouse, digitizer, high-resolution color display, and laser printer.

Solution:

- A preliminary statement of software scope can be developed: **[Bounded statements]**
- The mechanical CAD software will **accept 2D and 3D geometric data** from an engineer.
- The designer will **interact and control the CAD system through a user interface** that will exhibit characteristics of good human / machine interface design.
- All **geometric data and other supporting information** will be **maintained in a CAD database.**
- **Design analysis modules will be developed** to **produce the required output**, which will be displayed on a variety of graphics devices.
- The software will be **designed to control and interact** with **peripheral devices** that include a mouse, digitizer, laser printer, and plotter.

- Assume major software functions are identified after refinement.
- An estimation table is developed after the decomposition technique for LOC:

Function	Estimated LOC
User interface and control facilities (UICF)	2,300
Two-dimensional geometric analysis (2DGA)	5,300
Three-dimensional geometric analysis (3DGA)	6,800
Database management (DBM)	3,350
Computer graphics display facilities (CGDF)	4,950
Peripheral control function (PCF)	2,100
Design analysis modules (DAM)	8,400
Estimated lines of code	*33,200*

- **Question:**
- **How the estimated values came????**

Example: The range of LOC estimates for the 3DGA is optimistic : 4600 LOC; most likely : 6900 LOC; and pessimistic : 8600 LOC.

The expected value for the 3DGA **S = (Sopt + 4Sm + Spess) / 6**

= (4600 + 4 * 6900 + 8600) / 6 = **6800 LOC**

Each function is listed like above and 33200 will be total lines of code.

Assumption:

- A **review of historical data indicates** that the organizational average productivity for systems of **CAD type** is **620 LOC/pm.**
  - **i.e.,** 1 Guy – will write 620 lines of code in a month.
- Let us say Labor rate = $8000 / month,
  - The cost per line of code = $8000 / 620 → $13
- Based on LOC estimate and the historical productivity data,
  - The total estimated project **cost** is = $13 * 33200 = $431000 and
  - Estimated **effort** per month is = 33200 / 620 = 53.5 = 54 person months approximately.

Problem:

Suppose an online employment application is to be developed with the following services and estimated LOC for the services are: registration page 13 K, data entry 20 K, login page 5K, images/docs uploading 3K, and report generation 4K. Assuming that your organization produces 275 LOC/pm with a burdened labor rate of $3570 per person-month, estimate the effort and cost required to build the software using the LOC-based estimation technique.

- Organization produces 275 **LOC/pm i.e.,**
  - 1 Guy – will write 275 lines of code in a month.
- Labor rate = $3570 / month,
    - The cost per line of code = $3570 / 275 → $12.98 → $13
- Based on LOC estimate and the historical productivity data,
    - The total estimated project **cost** is = 13 * 45000 = $585000 and
    - Estimated **effort** per month is = 45000 / 275 = 163.63 = 164 person months approximately.
    - Total Cost = 164 * $3570 = $585480

Problem:

Assume that you are the project manager for a company that builds software for household robots. You have been contracted to build the software for a robot that mows the lawn for a homeowner. Write a statement of scope that describes the software. Be sure your statement of scope is bounded. If you're unfamiliar with robots, do a bit of research before you begin writing. Also, state your assumptions about the hardware that will be required. Alternate: Replace the lawn-mowing robot with another problem that is of interest to you. Do a functional decomposition of the robot software as described and Estimate the size of each function in LOC.

Assuming that your organization produces 450 LOC/pm with a burdened labor rate of $7000 per person-month, estimate the effort and cost required to build the software using the LOC-based estimation technique.

Answer:

- A company that build software for house – hold robotics.
- Assume that further refinement has occurred and that the major software functions are identified.
- Following the decomposition technique for LOC, the estimation table is developed.

Function	Estimated LOC
User interface and control facilities (UICF)	2,300
Two-dimensional geometric analysis (2DGA)	5,300
Three-dimensional geometric analysis (3DGA)	6,800
Database management (DBM)	3,350
Computer graphics display facilities (CGDF)	4,950
Peripheral control function (PCF)	2,100
Design analysis modules (DAM)	8,400
Estimated lines of code	33,200

Given:

- Organization produces **450 LOC/pm i.e.,**
  - 1 Guy – will write 450 lines of code in a month.
- Labor rate = $7000 / month,
  - The cost per line of code = $7000 / 450 → $15.55
- Based on LOC estimate and the historical productivity data,
  - The total estimated project **cost** is = 15.55 * 33200 = $516444 and
  - Estimated **effort** per month is = 33200 / 450 = 73.77 = ~74 person months.
  - Total Cost = 74 * $7000 = $518000

Problem Statement: Take the Library management system case. Software developed for library will accept data from operator for issuing and returning books. Issuing and returning will require some validity checks. For issue it is required to check if the member has already issued the maximum books allowed. In case for return, if the member is returning the book after the due date then fine has to be calculated. All the interactions will be through user interface. Other operations include maintaining database and generating reports at regular intervals.

VIT-AP UNIVERSITY

Major software functions identified.

1.User interface

2.Database management

3.Report generation

For user interface

Sopt : 1800

Sm : 2000

Spess : 4000

EV for user interface

EV = (1800 + 4*2000 + 4000) / 6

EV = 2300

For database management

Sopt : 4600

Sm : 6900

Spess : 8600

EV for database management

EV = (4600 + 4*6900 + 8600) / 6

EV = 6800

For report generation

Sopt : 1200

Sm : 1600

Spess : 3200

EV for report generation

EV = (1200 + 4*1600 + 3200) / 6

EV = 1800

Reconciling Estimates

VIT-AP UNIVERSITY

- The **results** gathered from the various estimation techniques **must be reconciled** to produce a single estimate of effort, project duration, and cost.
- If **widely divergent estimates** occur, **investigate the following causes**
  - The **scope** of the project is **not adequately understood** or has been misinterpreted by the planner
  - **Productivity data** used for problem-based estimation techniques is **inappropriate** for the application, obsolete (i.e., outdated for the current organization), or has been misapplied
  - The planner must **determine the cause** of divergence and then **reconcile** the estimates

Recap

Module – 5: Managing Software Projects

- Management Spectrum (4P's)
- People
  - Team Leaders & Important Characteristics
  - The Software Team - Toxic team environment and Solution
  - Team Organizations and Organization Paradigms
- Product
  - Determination of software scope and
  - Problem Decomposition
- Process - Melding the product and the process
- Project - Reasons for Project Fail and Solution
- Software Project Estimation - cost and effort estimation
  - Decomposition techniques – LOC / FP
  - **Empirical estimation models**

Empirical Estimation Models:

- Empirical estimation is a **technique or model** in which **empirically derived formulas** are **used for predicting the data** that are **required and essential part of the software project planning step**.
- These techniques are usually **based on**
  - **The data that is collected previously from a project** and **Some guesses**
  - **Prior experience** with the development of similar types of projects, and
  - Assumptions.
- It uses the **size of the software to estimate the effort.**

Empirical Estimation Models: The Structure of Estimation Models

- A typical estimation model is **derived using regression analysis on data collected from past software projects**
- The overall structure of such models takes the form

$$\mathbf{E = A + B * (e_v)^C}$$

where A, B, and C are empirically derived constants,

- E = Effort in person-months, and
- e_v = The estimation variable **(either LOC or FP)**

LOC-oriented estimation models

$E = 5.2 \times (KLOC)^{0.91}$	Walston-Felix model
$E = 5.5 + 0.73 \times (KLOC)^{1.16}$	Bailey-Basili model
$E = 3.2 \times (KLOC)^{1.05}$	Boehm simple model
$E = 5.288 \times (KLOC)^{1.047}$	Doty model for KLOC > 9

FP - Oriented models

$E = -91.4 + 0.355\ FP$	Albrecht and Gaffney model
$E = -37 + 0.96\ FP$	Kemerer model
$E = -12.88 + 0.405\ FP$	Small project regression model

Empirical Estimation Models: COCOMO

VIT-AP UNIVERSITY

- **The COCOMO (COnstructive COst MOdel)**
- Most popularly used software cost estimation models for a project
- It **estimates or predicts** the
  - **Effort required**
  - **Total project cost and**
  - **Scheduled time**
- Depends on the **number of lines of code (size)** for software product development.
- Developed by a software engineer **Barry Boehm in 1981**
- According to COCOMO, there are **three modes of software development projects that depends on complexity:**
    - **Organic**
    - **Semidetached**
    - **Embedded**

Empirical Estimation Models: COCOMO

Parameter	Organic	Semidetached	Embedded
Project size (Lines of Code - KLOC)	0 to 50	50 to 300	Above 300
Team Size	Simple & Small	Medium	Large
Domain Knowledge	Good	Medium	Large
Developer Experience	Experienced developers needed	Mix of Newbie and experienced developers	Good experience developers
Environment	Familiar	Less familiar	Unfamiliar, coupled with complex hardware
Innovation	Minor	Medium	Major
Deadline	Not tight	Medium	Very Tight
Example(s)	Inventory management system, Small Data Processing	New Operating System, Database Management System (DBMS), Compilers, and complex inventory management system,	Air Traffic Control, Banking software, ATM

Empirical Estimation Models: Basic COCOMO

- Basic COCOMO (Macro Estimation) which gives an initial quick and rough estimate of MM (man months) and development time.
- This model solely **based on lines of source code together with constant values** obtained from **software project types** rather than other factors which have major influences on the Software development process as a whole.

- Effort (E)= **$a(KLOC)^b$**
- Schedule Time or Time (D) = **$c(Effort)^d$**
- Person required or Avg. Resource Size = **Effort / Time**
- Productivity of Software **= KLOC/E**

Software Projects	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

- E = **Total effort** required for the project in **Man-Months (MM).**
- D = **Total time** required for project development in **Months (M).**
- KLOC = The size of the code for the project in **Kilo lines of code.**
- a, b, c, d = The constant parameters for a software project.

Empirical Estimation Models: COCOMO

Example: For a given project was estimated with a size of 300 KLOC. Calculate the Effort, Scheduled time for development, average resource size and productivity of all the three software project models.

Software Projects	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Organic:

- Effort (E)= 2.4* $(300)^{1.05}$ = **957.61 MM**
- Time (D) = 2.5*$(957.61)^{0.38}$ = **33.95 Months(M)**
- Average resource size = 957.61 / 33.95 = **28.21Mans ~29 Persons**
- Productivity of Software = KLOC/E = 300/957.61 = 0.3132 KLOC/MM = **313 LOC/MM**

Semidetached:

- Effort (E)=3.0 * $(300)^{1.12}$ = **1784.42 MM**
- Time (D) = 2.5*$(1784.42)^{0.35}$ = **34.35 Months(M)**
- Person Required (P) = E/D = 1784.42/34.35 = 51.9481 Persons ~52 Persons

Embedded:

- Effort (E)=3.6 * $(300)^{1.2}$ = **3379.46 MM**
- Time (D) = 2.5*$(3379.46)^{0.32}$ = **33.66 Months(M)**

Empirical Estimation Models: COCOMO

Example: For a given project was estimated with a size of 400 KLOC. Calculate the Effort, Scheduled time for development. Also, calculate the Average resource size and Productivity of all the software projects.

Software Projects	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Organic:

- Effort (E)= 2.4* $(400)^{1.05}$ = **1295.31 MM**
- Time (D) = 2.5*$(1295.31)^{0.38}$ = **38.07 Months(M)**
- Average resource size = 1295.31 / 38.07
- Productivity of Software = KLOC/E = 400/1295.31

Semidetached:

- Effort (E)=3.0 * $(400)^{1.12}$ = **2462.79 MM**
- Time (D) = 2.5*$(2462.79)^{0.35}$ = **38.45 Months(M)**

Embedded:

- Effort (E)=3.6 * $(400)^{1.2}$ = **4772.81 MM**
- Time (D) = 2.5*$(4772.81)^{0.32}$ = **38 Months(M)**

Empirical Estimation Models: COCOMO

Example: We have determined our project fits the characteristics of Semi-Detached mode. We estimate our project will have 32,000 delivered source instructions**.** Calculate the Effort, Scheduled time for development. Also, calculate the Average resource size and Productivity of the software project.

Software Projects	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Semidetached:

- Effort (E)=3.0 * $(32)^{1.12}$ = **146 MM**
- Time (D) = 2.5*$(146)^{0.35}$ = **14 Months(M)**
- Productivity = 32000 / 146 = 219 LOC / MM
- Person Required (P) = E/D = 146 / 14 = ~10 Persons

Empirical Estimation Models: COCOMO

VIT-AP UNIVERSITY

Problem

Suppose an online employment application is to be developed with the following services and estimated LOC for the services are: registration page 13 K, data entry 20 K, login page 5K, images/docs uploading 3K, and report generation 4K. Compute the Initial effort, Development Time, Productivity and Staff size for the above-mentioned project using suitable Basic COCOMO Model. Assume nominal modern programming practices, no tight project deadline, and high programmer capability.

Empirical Estimation Models: COCOMO

A company needs to develop digital signal processing software for one of its newest inventions. The software is expected to have 40000 lines of code. The company needs to determine the effort in person-months needed to develop this software using the basic COCOMO model. The multiplicative factor for this model is given as 2.8 for the software development on embedded systems, while the exponentiation factor is given as 1.20. What is the estimated effort in person-months?

- **Effort (E)**=$2.8 * (40)^{1.20}$ = **234.25 MM**

The number of function points of a proposed system is calculated as 500. Suppose that the system is planned to be developed in Java and the LOC/FP ratio of Java is 50. Estimate the effort (E) required to complete the project using the effort formula of basic COCOMO.

- Total function points = 500
- Each function point has an approximate lines of code = 50
- Total LOC = 500 * 50 = 25000
- Effort (E)=$2.5 * (25)^{1.0}$ **= 62.5**

Empirical Estimation Models: The Intermediate COCOMO

- The intermediate model **estimates** software development effort **in terms of size of the program** and other **related cost drivers parameters** (product parameter, hardware parameter, resource parameter, and project parameter) of the project.
- More detailed estimates for small to medium sized projects.

- Effort (E)= **$a(KLOC)^b$ *EAF MM**
- Schedule Time or Time (D) = **$c(Effort)^d$**
- Person required or Avg. Resource Size = **Effort / Time**
- **Productivity of Software = KLOC/E**

- E = **Total effort** required for the project in Man-Months (MM).
- D = **Total time** required for project development in Months (M).
- KLOC = The size of the code for the project in Kilo lines of code.
- a, b, c, d = The constant parameters for a software project.
- **EAF** = It is an **E**ffort **A**djustment **F**actor, which is calculated by **multiplying the parameter values of different cost driver parameters**. For ideal, the value is 1.

Empirical Estimation Models: The Intermediate COCOMO

COST DRIVERS PARAMETERS	VERY LOW	LOW	NOMINAL	HIGH	VERY HIGH
Product Parameter					
Required Software	0.75	0.88	1	1.15	1.4
Size of Project Database	NA	0.94		1.08	1.16
Complexity of The Project	0.7	0.85		1.15	1.3
Hardware Parameter					
Performance Restriction	NA	NA	1	1.11	1.3
Memory Restriction	NA	NA		1.06	1.21
Virtual Machine Environment	NA	0.87		1.15	1.3
Required Turnabout Time	NA	0.94		1.07	1.15
Personnel Parameter					
Analysis Capability	1.46	1.19	1	0.86	0.71
Application Experience	1.29	1.13		0.91	0.82
Software Engineer Capability	1.42	1.17		0.86	0.7
Virtual Machine Experience	1.21	1.1		0.9	NA
Programming Experience	1.14	1.07		0.95	NA
Project Parameter					
Software Engineering Methods	1.24	1.1	1	0.91	0.82
Use of Software Tools	1.24	1.1		0.91	0.83
Development Time	1.23	1.08		1.04	1.1

VIT-AP UNIVERSITY

Problem:

For a given project was estimated with a size of 300 KLOC. Calculate the Effort, Scheduled time for development by considering developer having very high application experience and very low experience in programming.

VIT-AP UNIVERSITY

Answer:

Given the estimated size of the project is: 300 KLOC

Developer having very high application experience: **0.82** (as per above table)

Developer having very low experience in programming: **1.14** (as per above table)

- **EAF = 0.82*1.14 = 0.9348**

- Effort (E)=3.0 * $(300)^{1.12}$ *0.9348 = **1668.07 MM**
- Time (D) = 2.5*$(1668.07)^{0.35}$ = **33.55 Months(M)**

Recap

Module – 5: Managing Software Projects

- Management Spectrum (4P's)
- People, Product, Process & Project
- Software Project Estimation - cost and effort estimation
  - Decomposition techniques – LOC / FP
  - Empirical estimation models
    - COCOMO (Constructive Cost Model)
    - COCOMO-II

Empirical Estimation Models: COCOMO II

- It is the model that **allows to estimate the cost, effort and schedule** when **planning a new software development activity.**
- **Three-level model** that allows **increasingly detailed estimates to be prepared as development progresses**
- **Application composition estimation model / Early Prototyping level** –
  - **Focuses** on applications which can be **quickly developed using interoperable components [Existing components]**
  - **Suitable** for projects built with Modern Graphical User Interface (GUI) builder tools.
  - Estimates based on **Object points** and a simple formula is used for effort estimation.
- **Early design stage model** –
  - Used once **requirements have been stabilized and basic software architecture has been established.**
  - Estimates based on **Function points** that are then translated to LOC.
- **Post-architecture stage model** –
  - Used **during the construction** of the software,
  - Estimates based on **LOC**.

Empirical Estimation Models: COCOMO II

- Requires **sizing information.**
- **Three different sizing options** are available as part of the model hierarchy**:**
  - **Object Points (OP)**
  - **Function Points (FP)**
  - **Lines of source Code (LOC)**
- **Application composition model / Early Prototyping Level**
  - Supports prototyping projects and projects where there is **extensive reuse**
  - Estimates effort in **object points/staff month**
  - **Object point:** An **indirect software measure** computed using counts of the number of
    1. Screens (at the user interface)
    2. Reports
    3. Components likely to be required to build the application

Empirical Estimation Models: COCOMO II

VIT-AP UNIVERSITY

Steps for effort estimation:

1. **Assess object counts** – Estimate the number of screens, reports, and 3GL components
2. **Classification** of complexity levels of each object
3. **Assign** complexity weights to each objects
4. **Determine Object Points : Multiplying** the original number of object instances by the weighting factor and **summing** to obtain a total object point count.
5. Compute **new object points (NOP)**
6. Calculate **Productive rate**, and compute the estimated **effort**.

Empirical Estimation Models: COCOMO II

Step-2: Classification of complexity levels: Classify each object instance into **Simple, Medium and Difficult** complexity levels depending on values of its characteristics.

For Screens

Number of views contained	*# and sources of data tables*		
	Total < 4 (< 2 server < 3 client)	*Total < 8 (2 – 3 server 3 – 5 client)*	*Total 8 + (> 3 server, > 5 client)*
< 3	Simple	Simple	Medium
3 – 7	Simple	Medium	Difficult
> 8	Medium	Difficult	Difficult

For Reports

Number of sections contained	*# and sources of data tables*		
	Total < 4 (< 2 server < 3 client)	*Total < 8 (2 – 3 server 3 – 5 client)*	*Total 8 + (> 3 server, > 5 client)*
0 or 1	Simple	Simple	Medium
2 or 3	Simple	Medium	Difficult
4 +	Medium	Difficult	Difficult

Empirical Estimation Models: COCOMO II

Step-3: Assign Complexity weights to each objects:

Object type	Complexity weight		
	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL component			10

- Weigh the number in each cell using the above scheme.
- Each object type is **classified into one of three complexity levels** (i.e., Simple, Medium, or Difficult).
- Complexity is a **function of the number and source of the client and server data tables** that are **required to generate the screen or report** and the **number of views or sections presented as part of the screen or report**.

- **Step-4:** Determine Object Points : **Multiplying** the original number of object instances by the weighting factor and **Summing** to obtain a total object point count.

Empirical Estimation Models: COCOMO Models

Step-5: New Object Points (NOP) = (Object points) x [(100 - %reuse)/100]

- When **component-based development or general software reuse is to be applied, the percent of reuse (%reuse)** is estimated and the object point count is adjusted.

Step-6: PRODuctivity rate (PROD) = NOP / person-month

- The productivity rate for different levels of **developer experience and development environment maturity** are:

Developer's experience/capability	Very low	Low	Nominal	High	Very high
Environment maturity/capability	Very low	Low	Nominal	High	Very high
PROD	4	7	13	25	50

Step-7: Estimated effort = NOP / PROD

Empirical Estimation Models: COCOMO Models

Use the COCOMO II model to estimate the effort required to build software that produces 10 screens, 8 reports, and will require approximately 70 software components. Assume that software has average complexity (screen 2, reports 5, 3gl components – 10) and average developer / environment maturity as 13. The system is component based development so percentage of reuse is 50%. Use the application composition model with object points.

Answer:

Object Type	Number	Complexity Weight - Average		Result
Screen	10	Average	2	20
Report	8	Average	5	40
3GL components	70	Average	10	700
			Object points	760

% reuse = 50

NOP = **(Object points) x [(100 - %reuse)/100]**

NOP = 760 * (100 – 50) / 100

= 380

What is PROD?

PROD = 13

Estimated effort = 380 / 13

= 29.23 pm

Empirical Estimation Models: COCOMO Models

Use the COCOMO II model to estimate NOP, Estimated Effort and Estimated Cost to build software for medium reservation system that produces 18 screens, 24 reports, and will require approximately 110 software components. Assume that software has medium complexity (screen 13, reports 17, 3gl components – 22). Assume very high developer / environment maturity. Past projects show burden labor rate of Rs. 2500PM. The system is component based development so percentage of reuse is 25%. Find

Answer:

Object Type	Number	Complexity Weight - Average		Result
Screen	18	Medium	13	234
Report	24	Medium	17	408
3GL components	110	Medium	22	2420
			Object points	3062

% reuse = 25

NOP = **(Object points) x [(100 - %reuse)/100]**

NOP = 3062 * (100 – 25) / 100

= 2296.5

What is PROD?

PROD = 50

Estimated effort = 2296.5 / 50

= 45.93 pm = ~46

Estimated cost = 46 * 2500

= Rs. 115000

Empirical Estimation Models: COCOMO Models

Use the COCOMO II model to estimate the effort required to build software for a simple ATM that produces 12 screens, 10 reports, and will require approximately 80 software components. Assume simple complexity and average developer / environment maturity. Use the application composition model with object points. The system is component based development so percentage of reuse is 20%.

Answer:

Object Type	Number	Complexity Weight - Simple		Result
Screen	12	Simple	1	12
Report	10	Simple	2	20
3GL components	80	NA		0
			Object Points	32

% reuse = 50

NOP = (Object Points) x [(100 - %reuse)/100]

NOP = OP * (100 - %reuse) /100
= 32 * (100 – 20)/100
= 25.6 object points

What is PROD?

PROD is given as average.

PROD = 13

Estimated effort = 25.6 / 13
= 1.96 pm
(person months)

Empirical Estimation Models: COCOMO Models

An airline application will have 3 screens (booking, pricing, availability) and will produce 1 report and The developers' experience is very low. The levels of difficulty of the pricing screen, the availability screen and the sales report are classified as simple, medium and medium, respectively. There is no 3GL component.

Assumption:

- A booking screen: Records a new ticket booking
- A pricing screen: Shows the rate for each day and each flight
- An availability screen: shows available flights
- A sales report: Shows total sale figures for the month and year, and compares figures with previous months and years

Name	Object Type	Number	Complexity Weight - Average		Result
Booking	Screen	1	Simple	1	1
Pricing	Screen	1	Simple	1	1
Availability	Screen	1	Medium	2	2
Sales	Report	1	Medium	5	5
				Object points	9

% reuse = 0

NOP = **9 x [(100 - 0)/100]**

= 9 * 100 / 100

= 9

What is PROD?

PROD = Very Low = 4

Estimated effort = 9 / 4

= 2.25pm

Empirical Estimation Models: COCOMO II

Problem: Consider a database application project with the following characteristics:

I. The application has 4 screens with 4 views each and 7 data tables for 3 servers and 4 clients.

II. The application may generate 2 reports of 6 sections each from 7 data tables for 2 server and 3 clients. There is 10% reuse of object points. The developer's experience and capability in the similar environment is low. The maturity of organization in terms of capability is also low. Calculate the object point count, New object points and effort to develop such project.

For Screens

Number of views contained	*# and sources of data tables*		
	Total < 4 (< 2 server < 3 client)	*Total < 8 (2 – 3 server 3 – 5 client)*	*Total 8 + (> 3 server, > 5 client)*
< 3	Simple	Simple	Medium
3 – 7	Simple	Medium	Difficult
> 8	Medium	Difficult	Difficult

For Reports

Number of sections contained	*# and sources of data tables*		
	Total < 4 (< 2 server < 3 client)	*Total < 8 (2 – 3 server 3 – 5 client)*	*Total 8 + (> 3 server, > 5 client)*
0 or 1	Simple	Simple	Medium
2 or 3	Simple	Medium	Difficult
4 +	Medium	Difficult	Difficult

Empirical Estimation Models: COCOMO II

Solution:

- This project comes under the category of application composition estimation model.
- Number of screens = 4 screens with 4 views each
- Number of reports = 2 reports with 6 sections each
- From Screens Table: Each screen will be of **medium (2)** complexity (7 data tables for 3 servers & 4 clients)
- From Reports Table: Each report will be **difficult (8)** complexity (7 data tables for 2 servers & 3 clients)

Object type	Complexity weight		
	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL component			10

Object Point = 4 * 2 + 2 * 8 = 24
Reuse = 10%
NOP = 24 * (100 – 10)/100 = 21.6
PROD = ??
= LOW value = 7
Effort = 21.6 / 7 = 3.086PM

Empirical Estimation Models: The Software Equation

- A **dynamic multivariable model** that assumes a **specific distribution of effort over the life of a software development project**
- Derived from productivity data collected for over 4000 contemporary software projects
- **$E = \{(LOC * B^{0.333}) / P^3\} * (1/t^4)$**

where

- E = Effort in person-months or person-years
- t = Project duration in months or years
- B = "Special skills factor"
- P = "Productivity parameter" that reflects: overall process maturity and management practices, the extent to which good software engineering practices are used, the level of programming languages used, the state of the software environment, the skills and experience of the software team, and the complexity of the application

Typical values for P =

- Development of real-time embedded software = **2000**
- Telecommunication and systems software = **10000**
- Business systems applications = **28000**
- The productivity parameter **can be derived for local conditions using historical data collected from past development efforts.**

- The software equation has two independent parameters:
  1. An estimate of size (in LOC) and
  2. An indication of project duration in calendar months or years.

Empirical Estimation Models: The Software Equation

- **B** increases slowly as "**the need for integration, testing, quality assurance, documentation, and management skills grows".**
  - For small programs (KLOC = 5 to 15), B = 0.16
  - For programs greater than 70 KLOC, B = 0.39
  - For programs (>15 & < 70) KLOC, B = (0.16 + 0.39) / 2 = 0.28

- **Simplification**
  - To simplify the estimation process and use a more common form for their estimation model,
  - **Putnam and Myers** suggest a set of equations derived from the software Equation
- **Minimum Development time (t_{min}) =**

  $t_{min} = 8.14 * (LOC/p)^{0.43}$ in months for $t_{min} > 6$ months

  $E = 180 * B \times t^3$ in person-months for E >= 20 person-months

  [t is represented as years]

Empirical Estimation Models: The Software Equation

- Let P = 12,000 (the recommended value for scientific software) for the CAD software having 33200 LOC, calculate t_{min} and E

Solution:

$t_{min} = 8.14 * (LOC/p^{0.43})$

$= 8.14 *(33200/12000)^{0.43} = 12.6$ Calendar Months

$E = 180 * B \times t^3$

$= 180 * 0.28 * (1.05)^3 = 58$ person-months

How 1.05 → 12.6 calendar months &

t needs to be expressed in years → 12.6 / 12

Empirical Estimation Models: The Software Equation

- Use the software equation to estimate the simple web development software having 48000 LOC and Assume that P = 8000

Answer:

- P = 8000, LOC = 48000
- t_{min} = 8.14 * (LOC/$P^{0.43}$) in months
  - t_{min} = 8.14 * (48000/8000)$^{0.43}$ = 8.14 * 1.08 = 8.79 calendar months
  - t = 0.7325 years
- E = 180 * B x t^3 in person-months
  =180 * 0.28 * $(0.7325)^3$
  = 19.80 person-months =~20 person-months

Practice Problems

- Design the effort and duration using the above details for basic COCOMO model. Given, Number of user inputs = 15 Number of user outputs = 3 Number of external interfaces = 11 1 function point = 20 LOC (as fourth generation language is used). Values of constant used in basic COCOMO model. a=2.4, b = 1.05, c = 2.5, d = 0.38

Project Scheduling : Roadmap to guide all project activities.

VIT-AP UNIVERSITY

- During CAT / FAT Exam Preparation
- Need to study 3 modules in a Day for a Course.
- You have the appropriate material, know what to do.
- Every student will have their own approach, started reading.
- After one hour - your father / friend called to know the status of preparation - completed 30% of syllabus
- After four hours - again father / friend called to know the status of preparation - completed 75% of syllabus
- After two hours - again father / friend called to know the status of preparation - completed 85% of syllabus, he said due to phone call from a friend had a small snag, he said he will get-out of it and back track in couple of minutes.
- father / friend "Did you cover all the modules by exam time?"
- "No problem," "I'm close to 90 percent complete." will complete by the time and will write exam.
- Finally finished the reading (with the help of others) just before the examination and wrote the exam very well.
- This story has been repeated tens of thousands of times **by students / software developers** during the past five to six decades. **The big question is why?**

Project Scheduling

VIT-AP UNIVERSITY

- Basic Concept – why late delivery happens?
- Project Scheduling
  - Basic Principles of project scheduling
  - The Relationship Between People and Effort
  - Effort Distribution
- Defining a Task Set for the Software Project
- Defining a Task Network
- Scheduling Process and Activities
- Earned Value Analysis

Project Scheduling

- Scheduling is the culmination of a planning activity that is a primary component of software project management.
- When combined with estimation methods and Risk , scheduling establishes a road map for the project manager
- One of the most difficult jobs for a project manager .
- Begins with process decomposition . It involves separating total work involved in a project into separate activities and judging the time required to complete these activities .
- Managers estimate time and resources required to complete activities and organize them into a coherent sequence .
- When estimating schedules, should NOT assume that every stage of the project will be problem-free.
  - People may fall ill or may leave
  - Hardware may break down
  - Essential software/hardware may be delivered late
- If the project is new and technically advanced, Then certain parts of it may turn out be more difficult and take longer than originally anticipated.

Project Scheduling – Why it is important

- Today's software systems are large, sophisticated and complex
  - Many software engineering tasks occur in parallel
  - Result of work performed during one task may have a profound effect on work conducted in another task.
  - Task interdependencies are very difficult to understand without a schedule
  - It is virtually impossible to assess progress on a large software project without a detailed schedule

Project Scheduling - Why late delivery happens?

Root Causes for Late Software

- An **Unrealistic deadline** established by someone outside the team and forced on managers & practitioners
- Changing customer requirements that are not reflected in schedule changes
- An honest underestimate of the amount of effort and / or the number of resources required to do the job.
- Predictable and unpredictable risks that were not considered
- Technical difficulties that could not have been predicted in advance
- Human difficulties that could have not been predicted in advance
- Miscommunication among project staff that results in delays
- Project Manager fails to track progress
  - Unaware that project is behind schedule and in trouble
  - Failure to take corrective action to correct problems

Project Scheduling - Why late delivery happens?

VIT-AP UNIVERSITY

How to deal With Unrealistic Schedule Demands:

- Perform a detailed estimate of effort and time using historical data.
- Using **an incremental process model,** develop a strategy to deliver critical functionality by deadline, **delay** other **requested functionality, document the plan.**
- **Meet** with the customer and explain why the deadline is unrealistic using estimates based on prior team performance.
- Offer an **incremental development and delivery strategy** as an alternative to increasing resources or allowing the schedule to slip beyond the deadline.

Project Scheduling - Software Project Scheduling Principles

- **Compartmentalization** - The project must be decomposed into a number of manageable activities, actions and tasks
- **Interdependency** - Tasks that can be completed in **parallel** must be **separated** from those that must completed **serially.**
- **Time allocation** - Each task must be allocated some no. of work units with start date and completion date that take the task interdependencies into account
- **Effort validation** - Be sure resources are available at any given time.
- **Defined Responsibilities** - Every task should be assigned to a specific team member
- **Defined outcomes** - Every task must have a defined outcome (Ex. work product or deliverable)
- **Defined milestones** - Tasks should be associated with a project milestones. A milestone is **accomplished** when one or more work products from an engineering task have passed quality review.

Project Scheduling - Relationship Between People and Effort

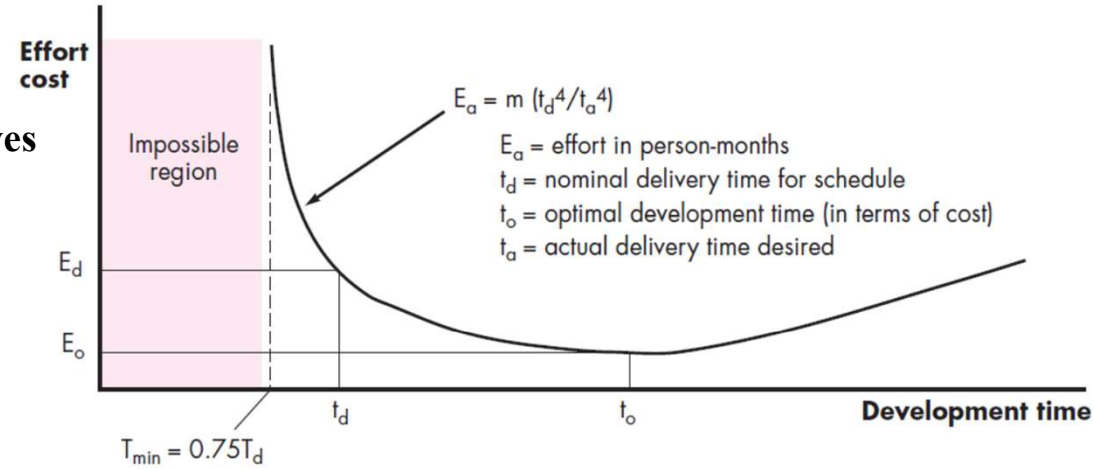
Common Myth: "If we fall behind schedule, add people to a project and catch up later in the project" Why does this not work.

- Often causing the schedules to slip further.
- New people has to train, learn the system – learning takes the time
- Teaching takes time away from product work - People who teach them are the same people who were doing the work.
- Increase in communication paths and complexity of communication
- Over the years, empirical data & theoretical analysis have demonstrated that project schedules are elastic
  - It is possible to compress a desired project completion date by adding additional resources to some extent
  - It is possible to extend completion date (by reducing resources).

Project Scheduling - Relationship Between People and Effort

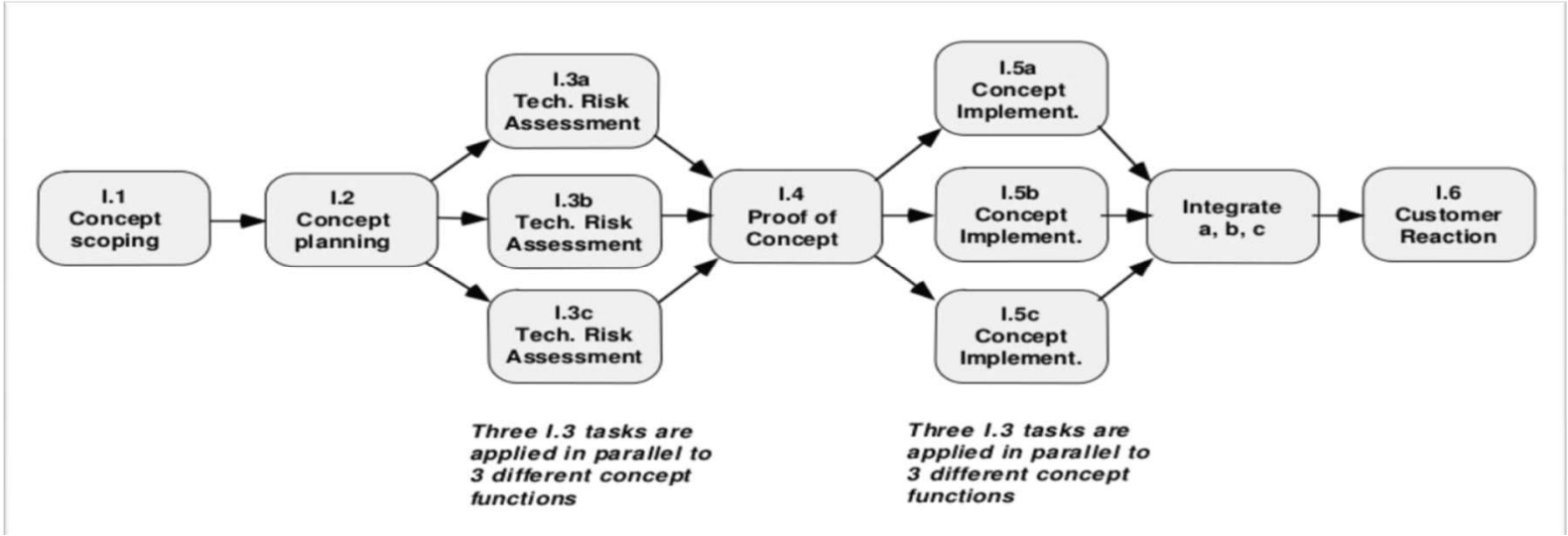
- The curve indicates a minimum value $\mathbf{t_o}$ that indicates the least cost for delivery (i.e., the delivery time that will result in the least effort expended).
  - As we move left of $\mathbf{t_o}$ (i.e., as we try to accelerate delivery), the curve rises nonlinearly.
- Assume that a project team has estimated a level of effort $\mathbf{E_d}$ will be required to achieve a nominal delivery time $\mathbf{t_d}$ that is optimal in terms schedule and available resources.
- Although it is possible to accelerate delivery, the curve rises very sharply to the left of $\mathbf{t_d}$.
- The PNR curve indicates that the project delivery time cannot be compressed much beyond $\mathbf{0.75t_d}$

- **If we attempt further compression, the project moves into "the impossible region" and risk of failure becomes very high.**
- The Putnam – Norden – Rayleigh (PNR) curve provides an indication of **effort applied** and **delivery time** for a software project.



Project Scheduling - Task Networks

1. A Task Network (Activity Network) is a graphic representation of the task flow for a project .
2. A task network depicts each software engineering task, its dependency on other tasks, and its projected duration.
   - The task network is used to compute the critical path, a timeline chart and a variety of project information
   - The task network is a useful mechanism for depicting intertask dependencies and determining the critical path.



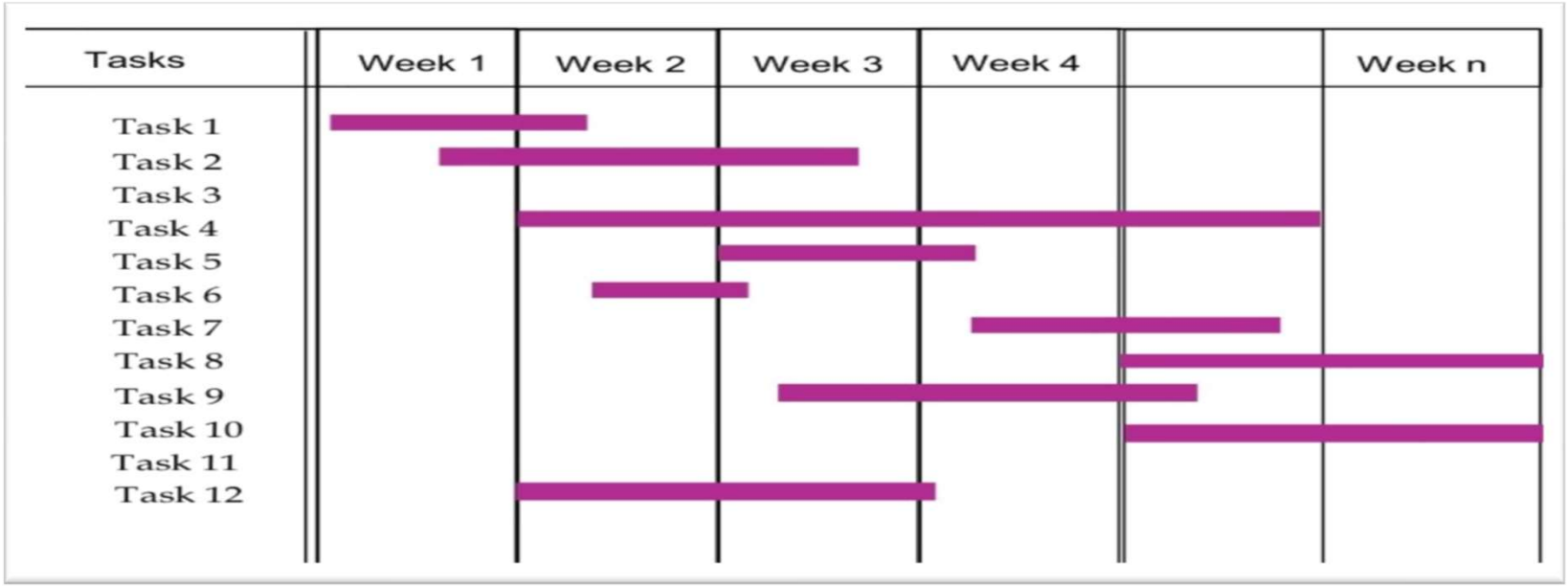
Project Scheduling - Scheduling Methods

1. Two project scheduling methods can be applied to software development :
    1. PERT: Program Evaluation and Review Technique
    2. CPM: Critical Path Method
2. Both techniques are driven by information already developed in earlier project planning activities :
    - Estimates of efforts
    - A decomposition of product function
    - Selection of appropriate process and task set
    - Decomposition of task
3. Both PERT and CPM provide quantitative tools that allows software planner to :
    - Determine critical path
    - The chain of tasks that determines project duration
    - Establish "most likely" time schedules for individual tasks
    - Calculate "boundary times"

Project Scheduling - Timeline charts / Gantt Chart

VIT-AP UNIVERSITY

1. When creating a software project schedule, the planner begins with a set of tasks. If automated tools are used, work breakdown is input as a task network. As a consequence, a timeline chart is generated.
2. Can be developed for the entire project. Also, separate charts can be developed for each project function .
3. Enables to determine what works will be conducted at a given point in time.



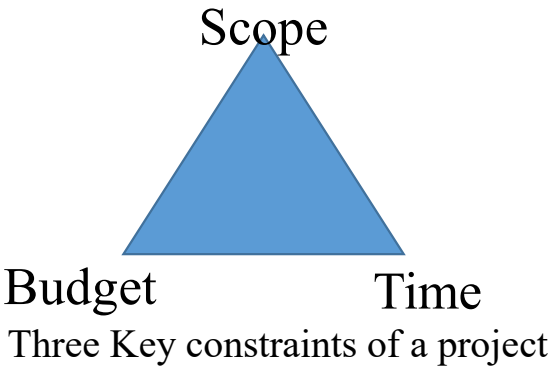
Project Scheduling - Project Effort Distribution

VIT-AP UNIVERSITY

- **How should effort be distributed across the software process workflow?**

- **The 40-20-40 rule (a rule of thumb):**
  - 40% Front-end activities – Customer communication, analysis, design, review and modification
  - 20% Construction activities - Coding
  - 40% Back-end activities - Testing
- **Generally accepted guidelines are:**
  - 02-03 % Planning
  - 10-25 % Requirements analysis
  - 20-25 % Design
  - 15-20 % Coding
  - 30-40 % Testing and debugging

Project Scheduling - Earned Value Analysis (EVA)



Three Key constraints of a project

- Project Manager evaluates project's triple constraint of scope, time and cost
- Key Questions
  - Is the project performing to budget?
  - Is the project on schedule to deliver the agreed scope?

- **Earned Value**
  - Used to measure **project health and project performance** (measure of progress)
  - Provides a quantitative indication of progress
  - Enables to assess the "percent of completeness" of a project using quantitative analysis
- Earned Value Analysis (EVA) is a quantitative technique for assessing progress.
- An industry standard way to
  - Measuring a projects progress at any given point in time
  - Forecast its completion date and final cost, and
  - Analyzing variances in schedule and budget as the project proceeds.

Project Scheduling - Earned Value Analysis (EVA)

VIT-AP UNIVERSITY

Earned Value Characteristics

- Point in Time Evaluation
- How much work did you PLAN to complete? (Planned Value (PV) or Budgeted Cost of Work Scheduled (BCWS))
- How much work did you ACTUALLY complete? (Earned Value (EV) or Budgeted Cost of Work Performed (BCWP))
- How much did you spend to complete the work? (Actual Cost (AC) or Actual Cost of Work Performed (ACWP))

Project Scheduling - Earned Value Analysis (EVA)

Example:

- A $10,000 software project is scheduled for 4 weeks. At the end of the third week, the project is 50% complete and the actual costs to date is $9,000
- What is Planned Value (PV), Earned Value (EV), Actual Cost (AC)?

- Planned Value (PV) = $7,500
- Earned Value (EV) = $5,000
- Actual Cost (AC) = $9,000

- Planned Value (PV) = $7,500 Total 4 weeks, at the end of 3 weeks.
- Earned Value (EV) = $5,000, 50% of work completed
- Actual Cost (AC) = $9,000, Spent

Project Scheduling - Earned Value Analysis (EVA)

VIT-AP UNIVERSITY

What is the project health?

- Planned Value (PV) = $7,500
- Earned Value (EV) = $5,000
- Actual Cost (AC) = $9,000

Schedule Variance (SV)

- SV = EV – PV = $5,000 – $7,500 = - $2,500

Schedule Performance Index (SPI)

- SPI = EV/PV = $5,000 / $7,500 = .66

Cost Variance (CV)

- CV = EV – AC = $5,000 - $9,000 = - $4,000

Cost Performance Index (CPI)

- CPI = EV/AC = $5,000 / $9,000 = .55

- Objective metrics indicate the project is behind schedule and over budget.
- On-target projects have an SPI and CPI of 1 or greater

Project Scheduling - Earned Value Analysis (EVA)

VIT-AP UNIVERSITY

Forecasting Costs

- If the project continues at the current performance, what is the true cost of the project?
- **Estimated Budget at Completion:**
  = Budget At Complete (BAC) / CPI
  = $10,000 / .55 = $18,181
  At the end of the project, the total project costs will be $18,181

Forecasting Schedule

- If the project continues at the current performance, what is the true schedule of the project?
- **Estimated Project duration:**
  = Planned duration / SPI
  = 4 / .66 = 6.06
  The project duration will be 6 weeks.

Project Scheduling - Earned Value Analysis (EVA)

- **Example:**
  - Budgeted cost of a project at $100000
  - Project to be completed in 10 months
  - After a month, you have completed 10 percent, total expense of $20000
  - Planned completion should have been about 20% at this point.
  - Find SV, SPI, CV and CPI, identify the Project Progress, Estimated Budget at Completion, and Estimated Project duration

Answer:

- BAC – Budgeted at Completion = $100000
- AC - Actual Cost – cost of the work that has been completed so far = $20000
- Calculate

Planned Value (PV) = Planned completion (%) * BAC
= 20 % * $100000 = $20000

Earned Value (EV) = Actual completion (%) * BAC
= 10 % * $100000 = $10000

Project Scheduling - Earned Value Analysis (EVA)

Cost Performance Index (CPI) = EV / AC

= $10000 / $20000 = 0.50

- Meaning : For every dollar spent producing 50% of the work only. [Not good idea]

Schedule Performance Index (SPI) = EV / PV

= $10000 / $20000 = 0.50 (30 minutes)

- Meaning : For every estimated hour of work, only half an hour doing job.
- Team is very behind and has some catching up to do.
- Ideally results to be 1, project is on time and on schedule.

Estimated Budget at Completion:

= Budget At Complete (BAC) / CPI

= $100000 / .50 = $200000

Estimated Project duration:

= Planned duration / SPI

= 10 / .50 = 20 months

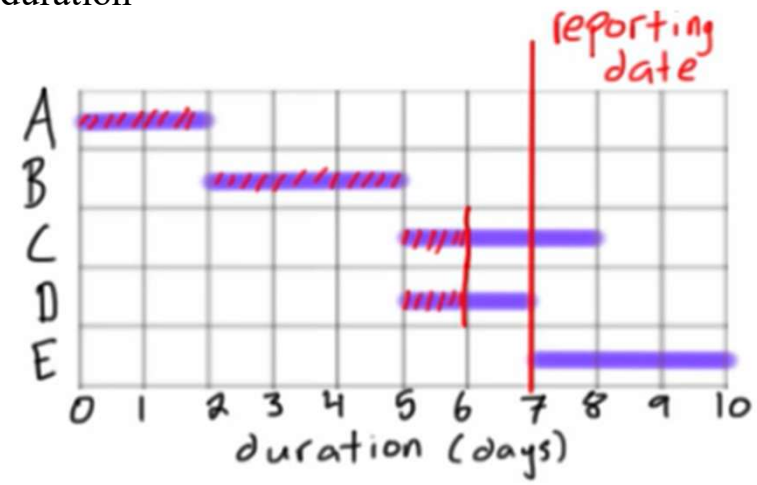
Project Scheduling - Earned Value Analysis

Example:

Activity	Predecessor	Duration (days)	Cost / Day	Total Cost
A	-	2	300	600
B	A	3	400	1200
C	B	3	400	1200
D	B	2	200	400
E	D	3	100	300

Field report at end of day 7		
Activity	Actual % Complete	Incurred Cost
A	100	600
B	100	1400
C	33	500
D	50	200
E	0	0

Find SV, SPI, CV and CPI, identify the Project Progress, Estimated Budget at Completion, and Estimated Project duration



Activity	AC	EV	PV	CPI	CV	SPI	SV
A				-	-	-	-
B				-	-	-	-
C				-	-	-	-
D				-	-	-	-
E				-	-	-	-
Total to date							

Project Scheduling - Earned Value Analysis

VIT-AP UNIVERSITY

Find SV, SPI, CV and CPI, identify the Project Progress, Estimated Budget at Completion, and Estimated Project duration

Activity	AC	EV	PV	CPI	CV	SPI	SV
A	600	600	600	-	-	-	-
B	1400	1200	1200	-	-	-	-
C	500	400	800	-	-	-	-
D	200	200	200	-	-	-	-
E	0	0	0	-	-	-	-
Total to date	**2700**	**2400**	**3000**	**0.889**	**-300**	**0.8**	**-600**

Estimated Budget at Completion:

= Budget At Complete (BAC) / CPI

= $3700 / .889 = 4161.98

Estimated Project duration:

= Planned duration / SPI

= 13 / .8 = 16.25days

Project Scheduling - Earned Value Analysis

Activity	Effort (days)	Cost per day	Total cost
1. Gather requirements	3	800	2,400
2. Create design	2	600	1,200
3. Build machine	4	900	3,600
4. Test and refine	4	700	2,800
5. Rollout	3	500	1,500

After 8 days

Activity	Incurred Cost	Actual % Complete
1. Gather requirements	2,400	100%
2. Create design	1,500	100%
3. Build machine	2,700	50%
4. Test and refine	700	25%
5. Rollout	0	0%

Activity	AC	EV	PV	CPI	CV	SPI	SV
1				-	-	-	-
2				-	-	-	-
3				-	-	-	-
4				-	-	-	-
5				-	-	-	-
Total							

Project Scheduling - Earned Value Analysis

Activity	AC	EV	PV	CPI	CV	SPI	SV
1	2400	2400	2400	-	-	-	-
2	1500	1200	1200	-	-	-	-
3	2700	1800	2700	-	-	-	-
4	700	700	700	-	-	-	-
5	0	0	0	-	-	-	-
Total	**7300**	**6100**	**7000**				

- CPI = EV / AC = 6100 / 7300 = 0.84
- CV = EV – AC = 6100 – 7300 = -1200
- SPI = EV / PV = 6100 / 7000 = 0.87
- SV = EV – PV = 6100 – 7000 = -900

Analysis:
CPI < 1 : Over Budgeted
CV = -1200, negative, Over Budgeted
SPI < 1 : Behind schedule
SV = -900, negative, Delay

Estimated Budget at Completion:
= Budget At Complete (BAC) / CPI
= 11500 / .84 = 13690.48

Estimated Project duration:
= Planned duration / SPI
= 16 / .87 = 18.40days

Project Scheduling - Earned Value Analysis

- Assume you are a software project manager and that you've been asked to compute earned value statistics for a small software project. The project has 56 planned work tasks that are estimated to require 582 person-days to complete. At the time that you've been asked to do the earned value analysis, 12 tasks have been completed. However the project schedule indicates that 15 tasks should have been completed. The following scheduling data (in person-days) are available:

Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Planned Effort	12.0	15.0	13.0	8.0	9.5	18.0	10.0	4.0	12.0	6.0	5.0	14.0	16.0	6.0	8.0
Actual Effort	12.5	11.0	17.0	9.5	9.0	19.0	10.0	4.5	10.0	6.5	4.0	14.5	-	-	-

Compute the SPI, schedule variance, percent scheduled for completion, percent complete, CPI, and cost variance for the project.

Project Scheduling - Earned Value Analysis

Schedule Performance Index(SPI) = BCWP / BCWS

SPI = sum of actual effort / sum of planned effort = 127.5 / 156.5 = 0.81

Schedule Variance (SV) = BCWP – BCWS = 127.5 – 156.5 = -29.0

Percent scheduled for completion = BCWS / BAC

=126.5 / 156.5 = 0.808 = 80.8%

Percent complete = BCWP / BAC

=127.5 / 156.5 = 0.8147 = 81%

Cost Performance Index CPI = BCWP / ACWP

=127.5 / 126.5 = 1.008 = 1.0

Cost Variance CV = BCWP - ACWP

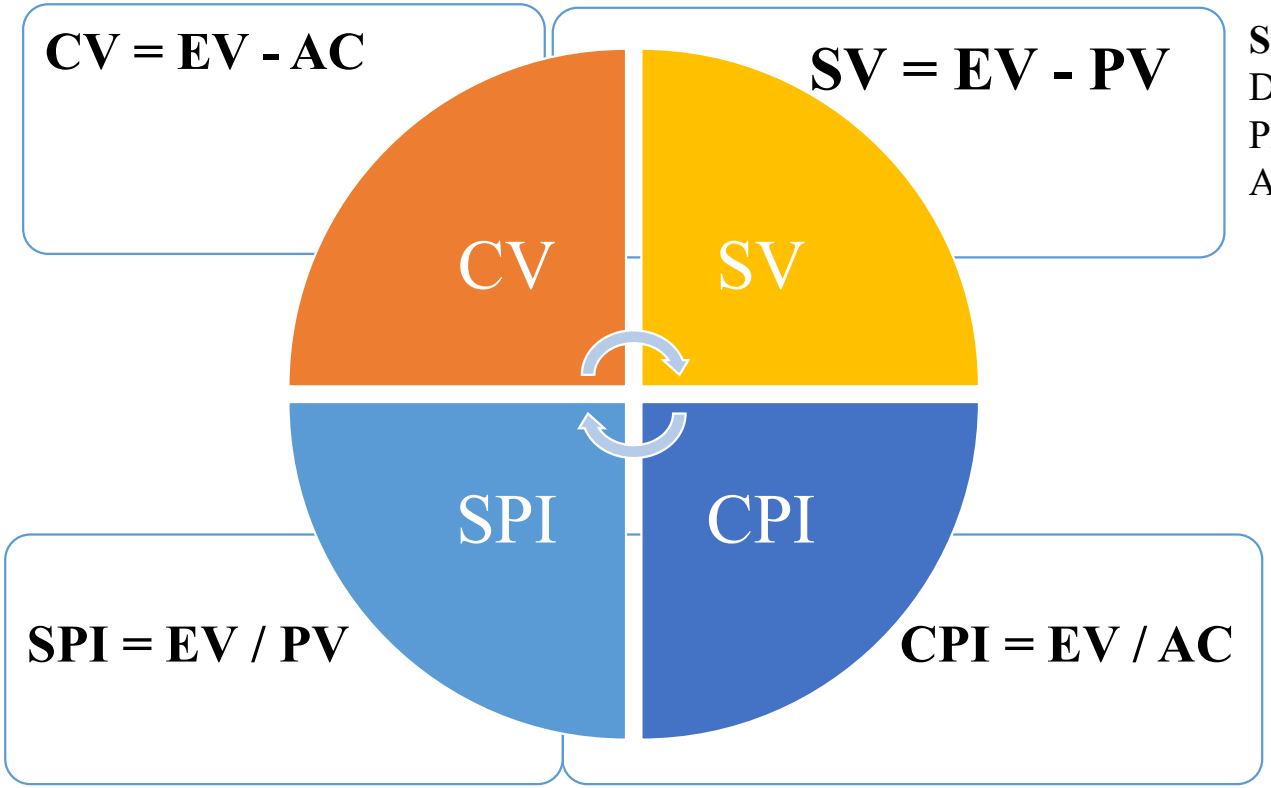
=127.5 - 126.5 = 1.0

Project Scheduling - Earned Value Analysis (EVA)

Indicators	Abbreviation	Definition
Budget at Completion	BAC	The total planned value of the project is the Budget At Completion
Duration	D	Project Duration
Planned Value Or Budgeted Costs of Work Schedule	PV or BCWS	Part of the approved cost estimate planned to be spent on the activity during a specific period.
Actual Cost or Actual Cost of Work Performed	AC Or ACWP	Total of costs supported in achieving work on the activity during a specific period
Earned Value or Budgeted Cost of Work Performed	EV or BCWP	Value of work completed at a given time

Project Scheduling - Earned Value Analysis (EVA)

Cost Variance:
Difference between Projected Cost and Actual Cost



Schedule Variance:
Difference between Projected schedule and Actual schedule

Schedule Performance Index:
A measure of how close the project is to performing the work as is was actually scheduled

Index Values:
< 1 : Over budget or behind schedule
> 1 : Under budget or ahead of schedule

Cost Performance Index:
A measure of how close the project is to spending on the work performed to what was planned to have been spent