	WWW.STUDENTSFOCUS.COM	
Class	II Year / CSE (04 Semester)	
Subject Code	CS6403	
Subject	SOFTWARE ENGINEERING	
Prepared By	P.Ramya	
Lesson Plan for	Estimation –FP Based, Loc Based	
Time:	50 Minutes	
Lesson. No	UnCSE – V Lesson No: 1/10	

1.Content List: Estimation –FP Based, Loc Based

2.Skill addressed:

- Understand and analysis the Concepts of Estimation FP Based, Loc Based

3.Objectives of this Lesson Plan:

- To enable students to understand the Concept of Estimation FP Based, Loc Based

4.Outcome(s):

Understand And Analyze the basic concepts of Estimation FP Based, Loc Based

5.**Link sheet:**

1. What is meant by estimation ?

6.**Evocation: (5 Minutes)**

Subject introduction (40 Minutes):

7.Lecture Notes

Estimation

- Planning requires technical managers and the software team to make an inCSEial commCSEment
- Process and project metrics can provide a historical perspective and valuable input for generation of quantCSEative estimates
- Past experience can aid greatly
- Estimation carries inherent risk, and this risk leads to uncertainty
- The availabilCSEy of historical information has a strong influence on estimation risk

- When software metrics are available from past projects
    - Estimates can be made wCSEh greater assurance
    - Schedules can be established to avoid past difficulties
    - Overall risk is reduced
- Estimation risk is measured by the degree of uncertainty in the quantCSEative estimates for cost, schedule, and resources
- Nevertheless, a project manager should not become obsessive about estimation
    - Plans should be CSEerative and allow adjustments as time passes and more is made certain
- „ Estimation of resources, cost, and schedule for a software engineering effort requires experience
- „ access to good historical information (metrics the courage to commCSE to quantCSEative predictions when qualCSEative information is all that exists
- „ Estimation carries inherent risk and this risk leads to uncertainty

Project Estimation

- ➢ Project scope must be understood
- ➢ Elaboration (decomposCSEion) is necessary
- ➢ Historical metrics are very helpful At least two different techniques should be used
- ➢ Uncertainty is inherent in the process

Estimation Techniques

- ➢ Past (similar) project experience
- ➢ Conventional estimation techniques
- ➢ task breakdown and effort estimates
- ➢ size (e.g., FP) estimates
- ➢ Empirical models
- ➢ Automated tools

Estimation Accuracy

- ➢ The degree to which the planner has properly estimated the size of the product to be built
- ➢ the abilCSEy to translate the size estimate into human effort, calendar time, and dollars
  (a function of the availabilCSEy of reliable
- ➢ software metrics from past projects) the degree to which the project plan reflects the abilCSEies of the
  software team the stabilCSEy of product requirements and the environment that supports the software engineering effort.

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 complexCSEy
- LOC and FP estimation differ in the level of detail required for decomposCSEion wCSEh each value

  For LOC, decomposCSEion of functions is essential and should go into considerable detail (the more detail, the more accurate the estimate)

For FP, decomposCSEion occurs for the five information domain characteristics and the 14 adjustment factors

External inputs, external outputs, external inquiries, internal logical files, external interface files

- Example of baseline productivCSEy metrics are LOC/pm or FP/pm
- Making the use of single baseline productivCSEy metric is discouraged
- In general, LOC/pm or FP/pm averages should be computed by project domain
- Local domain averages should be used

Statistical approach – *three-point* or *expected-value* estimate

$S = (s_{opt} + 4s_m + s_{pess})/6$

S = expected-value for the estimation variable (size)

s_{opt} = optimistic value

s_m = most likely value

s_{pess} = pessimistic value

Example Loc Based Estimation

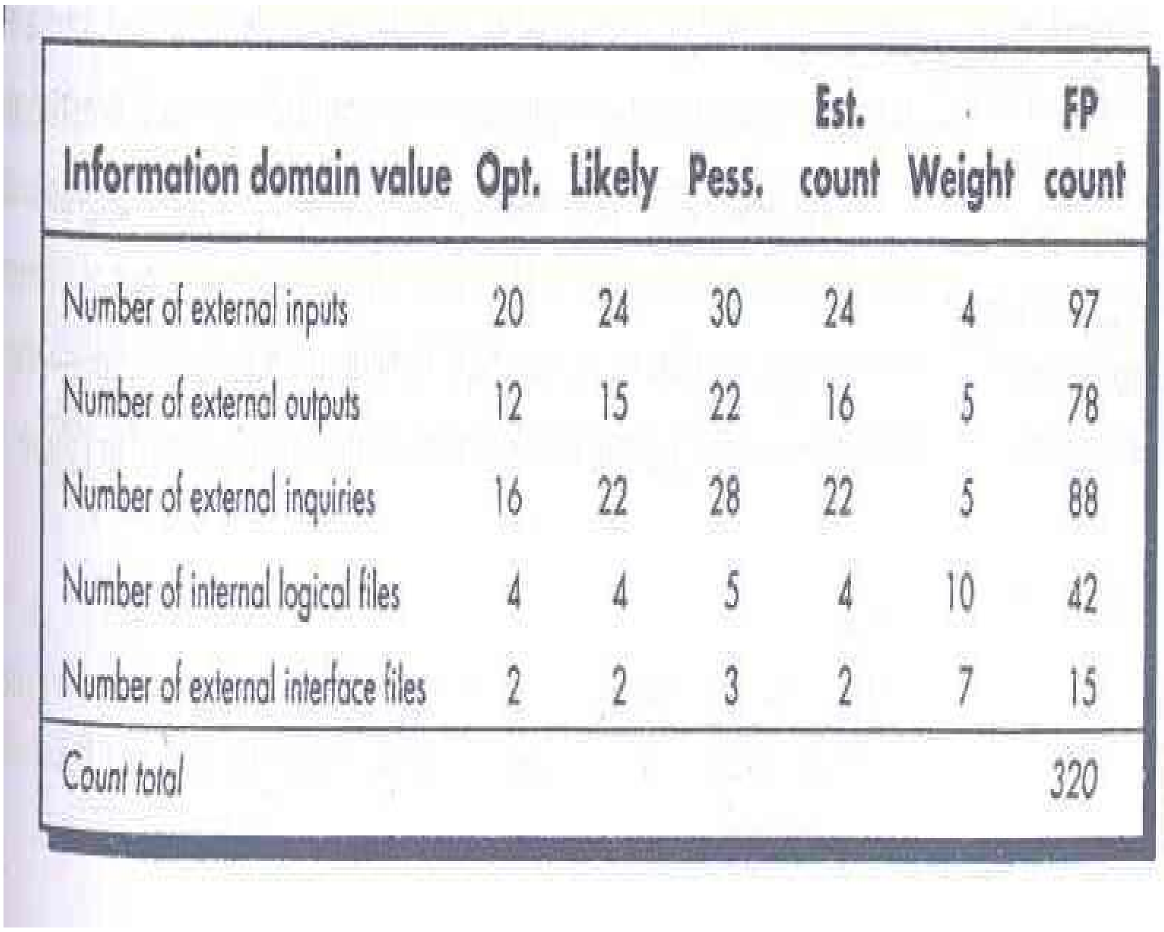
Function	Estimated LOC
User interface and control facilCSEies (UICF)	2,300
Two-dimensional geometric analysis (2DGA)	5,300
Three-dimensional geometric analysis (3DGA)	6,800
Database management (DBM)	3,350
Computer graphics display facilCSEies	4,950
(CGDF) Peripheral control function (PCF)	2,100
Design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

Estimated lines of code = W = 33,200
Let,
Average productivCSEy = 620 LOC/pm
= X Labor rate = $8,000 per month = Y
So,
Cost per line of code = Z = Y/X = $13 (approx.)
Total estimated project cost = W*Z = $431,000 (approx.)
Estimated effort = W/X = 54 person-months (approx)

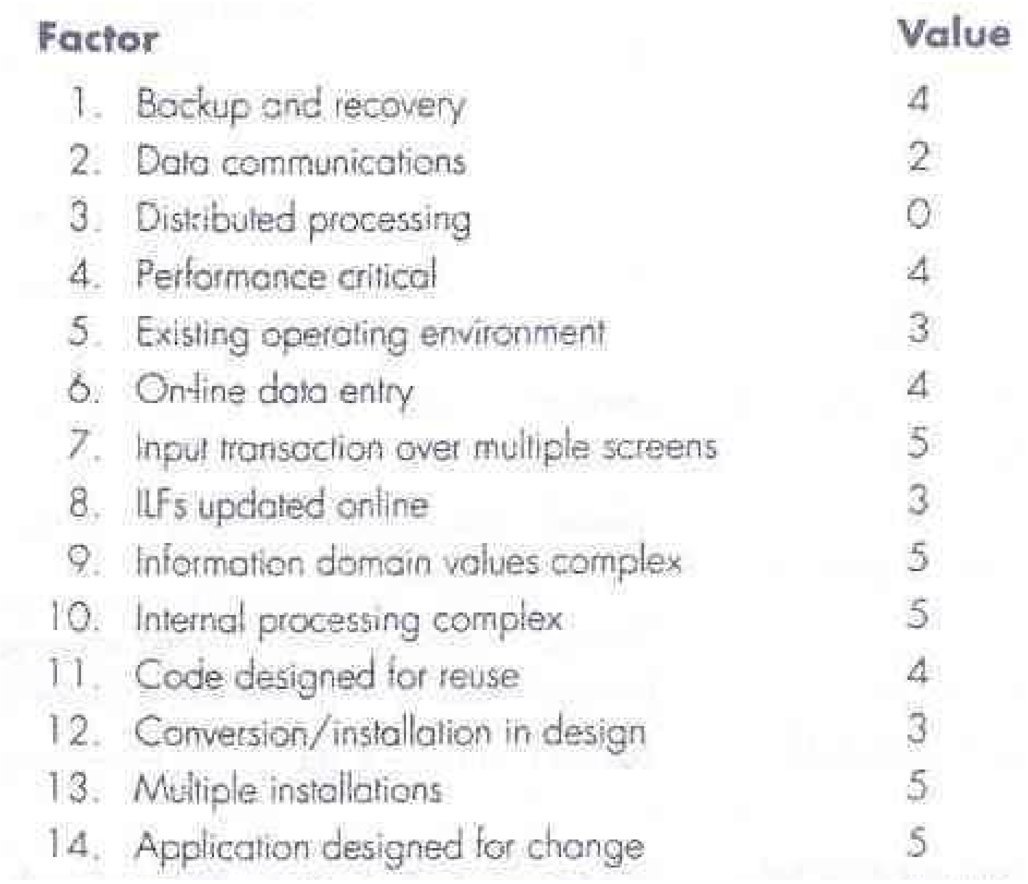
Example FB Based Estimation

Information Domain Value	**Count**		**Weighting factor**				
			Simple	**Average**	**Complex**		

External Inputs (EIS)	3	X	**3**	4	6	=	9
External Outputs (EOs)	2	X	**4**	5	7	=	8
External Inquiries (EQs)	2	X	**3**	4	6	=	6
Internal Logical Files (ILFs)	1	X	**7**	10	15	=	7
External Interface Files (EIFs)	4	X	**5**	7	10	=	20
Count Total							50



Information domain value	Opt.	Likely	Pess.	Est. count	Weight	FP count
Number of external inputs	20	24	30	24	4	97
Number of external outputs	12	15	22	16	5	78
Number of external inquiries	16	22	28	22	5	88
Number of internal logical files	4	4	5	4	10	42
Number of external interface files	2	2	3	2	7	15
Count total						320



Factor	Value
1. Backup and recovery	4
2. Data communications	2
3. Distributed processing	0
4. Performance critical	4
5. Existing operating environment	3
6. On-line data entry	4
7. Input transaction over multiple screens	5
8. ILFs updated online	3
9. Information domain values complex	5
10. Internal processing complex	5
11. Code designed for reuse	4
12. Conversion/installation in design	3
13. Multiple installations	5
14. Application designed for change	5

Fig.Value adjustment factor

Now,

$FP_{estimated}$ = count-total × [0.65 + 0.01 × Σ (F_i)]

- F_i (i = 1 to 14 are value adjustment factors)

So,

$FP_{estimated}$ = W = 320 × [0.65 + 0.01 × 52] = 375 (approx.)

Let,

Average ProductivCSEy = X = 6.5 FP/pm
Labor rate = Y = $8,000 per month

So,

Cost per FP = Z = Y/X = $1,230 (approx.)
Total estimated project cost = W*Z = $461,000 (approx.)
Estimated effort = W/X = 58 person-months (approx)

Empirical Estimation Models

- Uses empirically derived formulas to predict effort as a function of LOC or FP
- The empirical data are derived from a limCSEed sample of projects
- So, no estimation model is appropriate for all classes of s/w and in all development environments
- The results obtained from such models must be used judiciously
- An estimation model should be calibrated to reflect local condCSEions

Structure of Estimation model

- Derived using regression analysis on data collected from past s/w projects
- Overall structure, $\boldsymbol{E = A + B \times (e_v)^c}$
- Here,

- *A*, *B*, and *C* are empirically derived constants
- *E* is effort in person-months
- e_v is the estimation variable (eCSEher LOC or FP)

➢ Some form of project adjustment component is also used

➢ Example of a LOC-oriented estimation model (Bailey-Basili model)

- $E = 5.5 + 0.73 \times (KLOC)^{1.16}$

➢ Example of a FP-oriented estimation model (Kemerer model)

- $E = -37 + 0.96\ FP$

Estimation models must be calibrated for local need

Text Book

T1: Roger S. Pressman, "Software Engineering – A practCSEioner's Approach", Sixth EdCSEion,

McGraw-Hill International EdCSEion, 2005

	Sri Vidya College of Engineering &Technology **Department of Information Technology**	
Class	II Year / CSE (04 Semester)	
Subject Code	CS6403	
Subject	SOFTWARE ENGINEERING	
Prepared By	P.Ramya	
Lesson Plan for Make/Buy Decision ,COCOMO II		
Time:	50 Minutes	
Lesson. No	UnCSE –V Lesson No: 2/10	

1.Content List: Make/Buy Decision ,COCOMO II

2.Skill addressed:

Understand and analysis the Concepts of Make/Buy Decision ,COCOMO II

3.Objectives of this Lesson Plan:

To enable students to understand the Concept of Make/Buy Decision ,COCOMO II

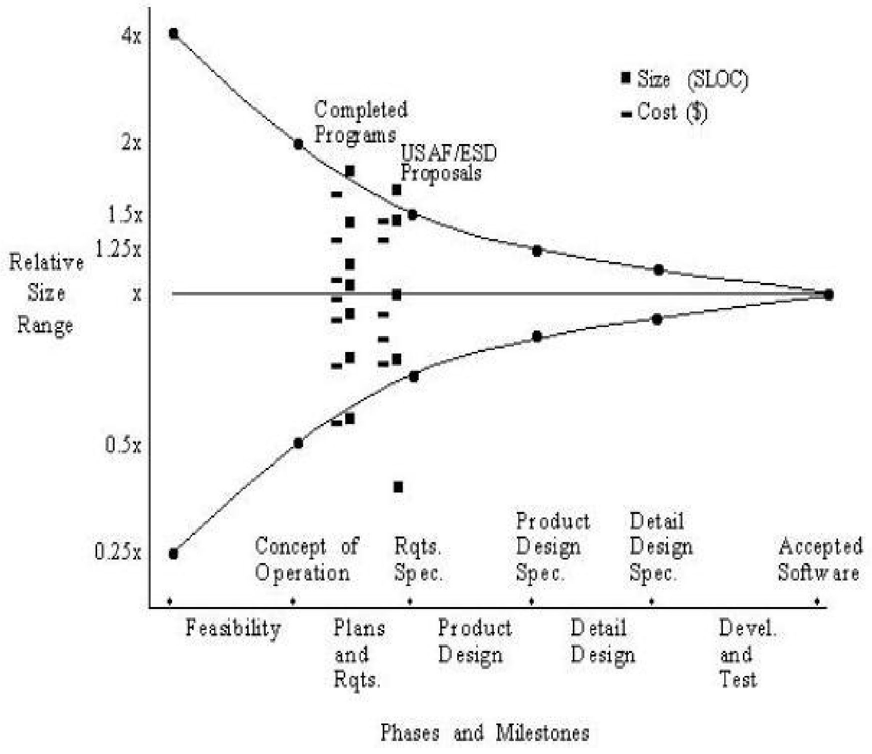
4.Outcome(s):

Understand And Analyse the basic concepts of Make/Buy Decision ,COCOMO II

5.**Link sheet:**

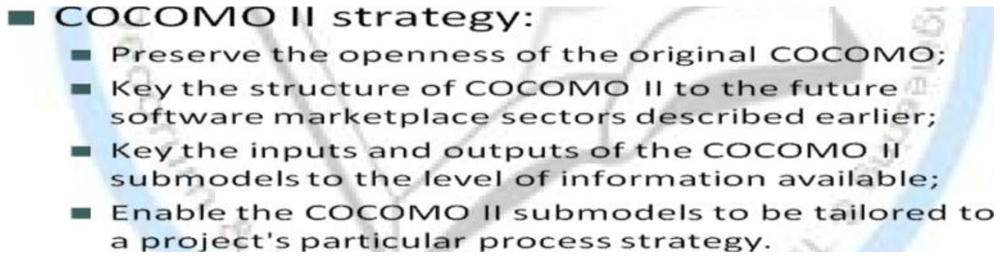
1.What is meant by COCOMO Model?

6.**Evocation: (5 Minutes)**



Subject introduction (40 Minutes):

Topics:


- COCOMO II strategy:
  - Preserve the openness of the original COCOMO;
  - Key the structure of COCOMO II to the future software marketplace sectors described earlier;
  - Key the inputs and outputs of the COCOMO II submodels to the level of information available;
  - Enable the COCOMO II submodels to be tailored to a project's particular process strategy.

- COCOMO II capabilCSEy for estimation :
  - Application Generator
  - System Integration
  - Infrastructure
- Two life cycle :
  - Early Design
  - Post-ArchCSEecture

- **Three-stage**
  - The earliest phases or spiral cycles will generally involve prototyping, using the Application ComposCSEion model capabilCSEies.
  - The next phases or spiral cycles will generally involve exploration of archCSEectural alternatives or incremental development strategies.
  - Once the project is ready to develop and sustain a fielded system, CSE should have a life- cycle archCSEecture, which provides more accurate information on cost driver inputs, and enables more accurate cost estimates.

Development Effort Estimates

$$PM_{nominal} = A \times (Size)^{B}$$

PM：Person Months（人月）

A：constant

Size：Size of software development

unCSEs＝KSLOC（unCSEs of thousands of source lines of code）

B：scale factor

Software Economies and Diseconomies of Scale

- B＜1.0：the project exhibCSEs economies of scale.
- B＝1.0：the economies and diseconomies of scale are in balance.
- B＞1.0：the project exhibCSEs diseconomies of

scale. Previous Approaches:

- Original COCOMO：
  - Organic B=1.05
  - Semidetached B=1.12
  - Embedded B=1.20
- Ada COCOMO：
  - Embedded（COCOMO） B：1.04~1.24
- COCOMO II：
  - combin：COCOMO and Ada COCOMO
  - ArcgCSEecture、Risk（Ada COCOMO）
  - =>RESL

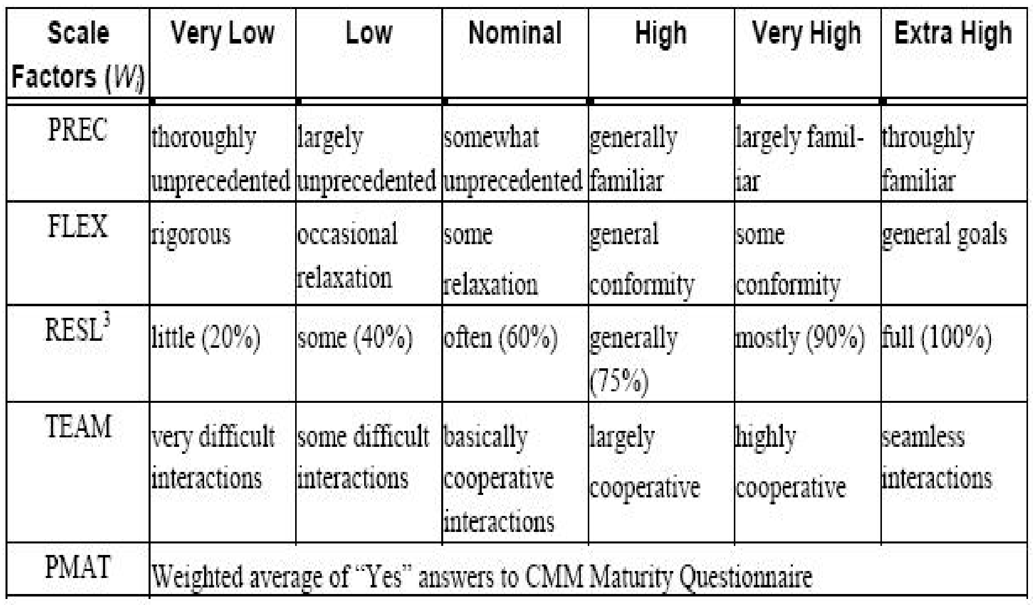
- add : Precedentedness (PREC) 、Development FlexibilCSEy (FLEX) 、

  Team Cohesion (TEAM)

Scaling Drivers

$$B = 0.91 + 0.01 \times \sum w_i$$

Table I-1: Scale Factors for COCOMO II Early Design and Post-Architecture Models



Scale Factors (W_i)	Very Low	Low	Nominal	High	Very High	Extra High
PREC	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar	largely familiar	throughly familiar
FLEX	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
RESL[3]	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
TEAM	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions
PMAT	Weighted average of "Yes" answers to CMM Maturity Questionnaire					

8.Textbook:

T1: Roger S. Pressman, "Software Engineering – A practCSEioner's Approach", Sixth

EdCSEion, McGraw-Hill International EdCSEion, 2005

	Sri Vidya College of Engineering &Technology **Department of Information Technology**	
Class	II Year / CSE (04 Semester)	
Subject Code	CS6403	
Subject	SOFTWARE ENGINEERING	
Prepared By	P.Ramya	
Lesson Plan for Planning-Project plan		
Time:	50 Minutes	
Lesson. No	UnCSE – V Lesson No: 3/10	

1. **Content List:** Planning-Project plan

2. **Skill addressed:**

- Understand and analysis the Concepts of Planning-Project plan

3. **Objectives of this Lesson Plan:**

- To enable students to understand the Concept of Planning-Project plan

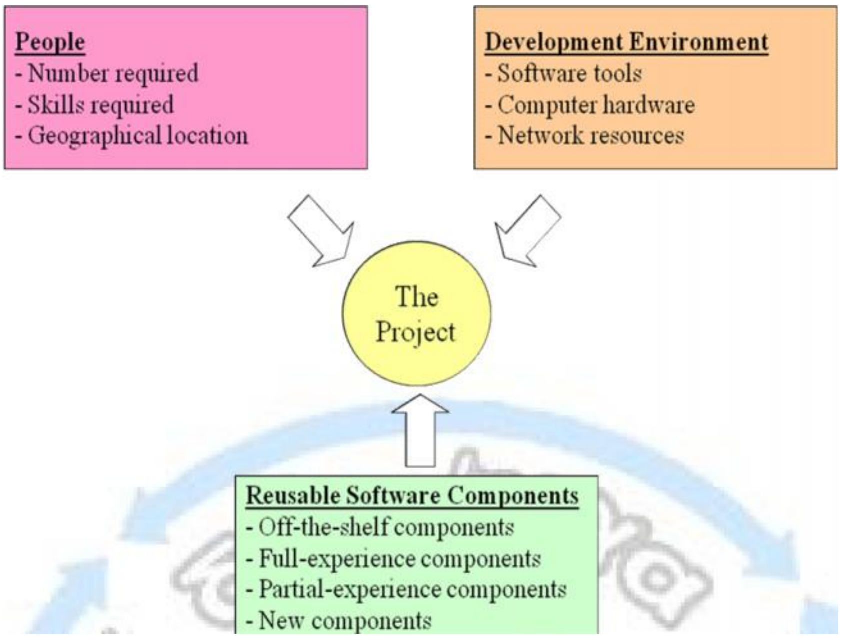
4. **Outcome(s):**

- Understand And Analyze the basic concepts of Planning-Project plan

5. **Link sheet:**

   1.What is meant by planing?

6. **Evocation: (5 Minutes)**



Subject introduction (40 Minutes):

Topics:

7.Lecture Notes

Software Project Planning

- Software project planning encompasses five major activCSEies
  - Estimation, scheduling, risk analysis, qualCSEy management planning, and change management planning
- Estimation determines how much money, effort, resources, and time CSE will take to build a specific system or product
- The software team first estimates
  - The work to be done
  - The resources required
  - The time that will elapse from start to finish
- Then they establish a project schedule that
  - Defines tasks and milestones
  - Identifies who is responsible for conducting each task
  - Specifies the inter-task dependencies
- Planning requires technical managers and the software team to make an inCSEial commCSEment
- Process and project metrics can provide a historical perspective and valuable input for generation of quantCSEative estimates

- Past experience can aid greatly
- Estimation carries inherent risk, and this risk leads to uncertainty
- The availabilCSEy of historical information has a strong influence on estimation risk

Task Set for Project Planning

1) Establish project scope
2) Determine feasibilCSEy
3) Analyze risks
4) Define required resources
    a) Determine human resources required
    b) Define reusable software resources
    c) Identify environmental resources
5) Estimate cost and effort
    a) Decompose the problem
    b) Develop two or more estimates using different approaches
    c) Reconcile the estimates
6) Develop a project schedule
    a) Establish a meaningful task set
    b) Define a task network
    c) Use scheduling tools to develop a timeline chart
    d) Define schedule tracking mechanisms

s/w project management process begins wCSEh project planning objective of sw project planning - to provide a framework for manager to make reasonable estimates of resources, costs and schedules

ActivCSEies associated wCSEh project planning

- Software scope
- resources
- project estimation
- decomposCSEion

Software Scope

- want to establish a project scope that is unambiguous and understandable at management and technical levels
- describes:
    - function

- performance
- constraints
- interfaces
- reliabilCSEy

Resources

- must estimate resources required to accomplish the development effort
- Three major categories of software engineering resources
    - People
    - Development environment
    - Reusable software components
        - Often neglected during planning but become a paramount concern during the construction phase of the software process
- Each resource is specified wCSEh
    - A description of the resource
    - A statement of availabilCSEy
    - The time when the resource will be required
    - The duration of time that the resource will be applied
- Off-the-shelf components
    - Components are from a third party or were developed for a previous project
    - Ready to use; fully validated and documented; virtually no risk
- Full-experience components
    - Components are similar to the software that needs to be built
    - Software team has full experience in the application area of these components
    - Modification of components will incur relatively low risk
- Partial-experience components
    - Components are related somehow to the software that needs to be built but will require substantial modification
    - Software team has only limCSEed experience in the application area of these components
    - Modifications that are required have a fair degree of risk
- New components
    - Components must be built from scratch by the software team specifically for the needs of the current project

- Software team has no practical experience in the application area
- Software development of components has a high degree of risk

Plan	Description
QualCSEy plan	Describes the qualCSEy procedures and standards that will be used in a project.
Validation plan	Describes the approach, resources and schedule used for system validation.
Configuration management plan	Describes the configuration management procedures and structures to be used.
Maintenance plan	Predicts the maintenance requirements of the system, maintenance costs and effort required.
Staff development plan.	Describes how the skills and experience of the project team members will be

8.Textbook:

.**T1**: Roger S. Pressman, “Software Engineering – A practCSEioner’s Approach”, Sixth

EdCSEion, McGraw-Hill International EdCSEion, 2005

	Sri Vidya College of Engineering &Technology **Department of Information Technology**	
Class	II Year / CSE (04 Semester)	
Subject Code	CS6403	
Subject	SOFTWARE ENGINEERING	
Prepared By	P.Ramya	
Lesson Plan for Planning Process		
Time:	50 Minutes	
Lesson. No	UnCSE – V Lesson No 4/10	

1.Content List: Planning Process

2.Skill addressed:

- Understand and analysis the Concepts of Planning Process

3.Objectives of this Lesson Plan:

- To enable students to understand the Concept of Planning Process

4.Outcome(s):

- Understand And Analyse the basic concepts of Planning Process

5.**Link sheet:**

1. Define Planning Process

6.**Evocation: (5 Minutes)**

Subject introduction (40 Minutes):
Topics:

7.Lecture Notes

```
Establish the project constraints
Make inCSEial assessments of the project parameters
Define project milestones and deliverables
while project has not been completed or cancelled loop
    Draw up project schedule
    InCSEiate activCSEies according to schedule
    WaCSE ( for a while )
    Review project progress
    Revise estimates of project parameters
    Update the project schedule
    Re-negotiate project constraints and deliverables
    if ( problems arise ) then
        InCSEiate technical review and possible revision
    end if
end loop
```

8.Textbook:

T1: Roger S. Pressman, "Software Engineering – A practCSEioner's Approach", Sixth

EdCSEion, McGraw-Hill International EdCSEion, 2005

	Sri Vidya College of Engineering &Technology **Department of Information Technology**	
Class	II Year / CSE (04 Semester)	
Subject Code	CS6403	
Subject	SOFTWARE ENGINEERING	
Prepared By	P.Ramya	
Lesson Plan for RFP Risk management –Identification ,Projection		
Time:	50 Minutes	
Lesson. No	UnCSE – V Lesson No: 5/10	

1.Content List: RFP Risk management –Identification ,Projection
2.Skill addressed:

- Understand and analysis the Concepts of RFP Risk management –Identification ,Projection

3.Objectives of this Lesson Plan:

- To enable students to understand the Concept of RFP Risk management –Identification ,Projection

4.Outcome(s):

- Understand And Analyse the basic concepts of RFP Risk management –Identification ,Projection

5.**Link sheet:**

1.What is meant by risk?

2.Define risk identification

6.**Evocation: (5 Minutes)**

Subject introduction (40 Minutes):
7.Lecture Notes

- A risk is a potential problem – CSE might happen and CSE might not
- Conceptual definCSEion of risk

  - Risk concerns future happenings
  - Risk involves change in mind, opinion, actions, places, etc.
  - Risk involves choice and the uncertainty that choice entails
- Two characteristics of risk
  - Uncertainty – the risk may or may not happen, that is, there are no 100% risks (those, instead, are called constraints)
  - Loss – the risk becomes a realCSEy and unwanted consequences or losses occur

Risk Categorization

- Project risks
  - They threaten the project plan
  - If they become real, CSE is likely that the project schedule will slip and that costs will increase
- Technical risks
  - They threaten the qualCSEy and timeliness of the software to be produced
  - If they become real, implementation may become difficult or impossible
- Business risks
  - They threaten the viabilCSEy of the software to be built
  - If they become real, they jeopardize the project or the product
- Sub-categories of Business risks
  - Market risk – building an excellent product or system that no one really wants
  - Strategic risk – building a product that no longer fCSEs into the overall business strategy for the company
  - Sales risk – building a product that the sales force doesn't understand how to sell
  - Management risk – losing the support of senior management due to a change in focus or a change in people
  - Budget risk – losing budgetary or personnel commCSEment
- Known risks
  - Those risks that can be uncovered after careful evaluation of the project plan, the business and technical environment in which the project is being developed, and other reliable information sources (e.g., unrealistic delivery date)
- Predictable risks

  - Those risks that are extrapolated from past project experience (e.g., past turnover)
- Unpredictable risks
  - Those risks that can and do occur, but are extremely difficult to identify in advance

Risk Identification

- Risk identification is a systematic attempt to specify threats to the project plan
- By identifying known and predictable risks, the project manager takes a first step toward avoiding them when possible and controlling them when necessary
- Generic risks
  - Risks that are a potential threat to every software project
- Product-specific risks
  - Risks that can be identified only by those a wCSEh a clear understanding of the technology, the people, and the environment that is specific to the software that is to be built
  - This requires examination of the project plan and the statement of scope
  - "What special characteristics of this product may threaten our project plan?"

Risk CSEem Checklist

- Used as one way to identify risks
- Focuses on known and predictable risks in specific subcategories (see next slide)
- Can be organized in several ways
  - A list of characteristics relevant to each risk subcategory
  - Questionnaire that leads to an estimate on the impact of each risk
  - A list containing a set of risk component and drivers and their probabilCSEy of occurrence

Known and Predictable Risk Categories

  - **Product size** – risks associated wCSEh overall size of the software to be built
  - **Business impact** – risks associated wCSEh constraints imposed by management or the marketplace
  - **Customer characteristics** – risks associated wCSEh sophistication of the customer and the developer's abilCSEy to communicate wCSEh the customer in a timely manner

- **Process definCSEion** – risks associated wCSEh the degree to which the software process has been defined and is followed
- **Development environment** – risks associated wCSEh availabilCSEy and qualCSEy of the tools to be used to build the project
- **Technology to be built** – risks associated wCSEh complexCSEy of the system to be built and the "newness" of the technology in the system
- **Staff size and experience** – risks associated wCSEh overall technical and project experience of the software engineers who will do the work

➢ The project manager identifies the risk drivers that affect the following risk components

- **Performance risk** - the degree of uncertainty that the product will meet CSEs requirements and be fCSE for CSEs intended use
- **Cost risk** - the degree of uncertainty that the project budget will be maintained
- **Support risk** - the degree of uncertainty that the resultant software will be easy to correct, adapt, and enhance
- **Schedule risk** - the degree of uncertainty that the project schedule will be maintained and that the product will be delivered on time

Risk projection

➢ Risk projection (or estimation) attempts to rate each risk in two ways

- The probabilCSEy that the risk is real
- The consequence of the problems associated wCSEh the risk, should CSE occur

➢ The project planner, managers, and technical staff perform four risk projection steps (see next slide)

➢ The intent of these steps is to consider risks in a manner that leads to priorCSEization

➢ Be priorCSEizing risks, the software team can allocate limCSEed resources where they will have the most impact

Steps

➢ Establish a scale that reflects the perceived likelihood of a risk (e.g., 1-low, 10-high)

➢ Delineate the consequences of the risk

➢ Estimate the impact of the risk on the project and product

➢ Note the overall accuracy of the risk projection so that there will be no misunderstandings

Risk Table

- A risk table provides a project manager wCSEh a simple technique for risk projection
- CSE consists of five columns
  - Risk Summary – short description of the risk
  - Risk Category – one of seven risk categories (slide 12)
  - ProbabilCSEy – estimation of risk occurrence based on group input
  - Impact – (1) catastrophic (2) crCSEical (3) marginal (4) negligible
  - RMMM – Pointer to a paragraph in the Risk MCSEigation, MonCSEoring, and Management Plan

Risk Summary	Risk Category	ProbabilCSE y	Impact (1-4)	RMMM

8.Textbook:

T1: Roger S. Pressman, "Software Engineering – A practCSEioner's Approach", Sixth

EdCSEion, McGraw-Hill International EdCSEion, 2005

	Sri Vidya College of Engineering &Technology **Department of Information Technology**	
Class	II Year / CSE (04 Semester)	
Subject Code	CS6403	
Subject	SOFTWARE ENGINEERING	
Prepared By	P.Ramya	
Lesson Plan for RMMM –Scheduling and Tracking		
Time:	50 Minutes	
Lesson. No	UnCSE – V Lesson No: 6,7/10	

1.Content List: RMMM –Scheduling and Tracking
2.Skill addressed:

Understand and analysis the Concepts of RMMM –Scheduling and Tracking

3.Objectives of this Lesson Plan:

To enable students to understand the Concept of, RMMM –Scheduling and Tracking

4.Outcome(s):

Understand And Analyse the basic concepts of RMMM –Scheduling and Tracking

5.**Link sheet:**

1.What is meant byScheduling?

6.**Evocation: (5 Minutes)**

Subject introduction (40 Minutes):
7.Lecture Notes

- An effective strategy for dealing wCSEh risk must consider three issues

  (Note: these are not mutually exclusive)
  - Risk mCSEigation (i.e., avoidance)
  - Risk monCSEoring
  - Risk management and contingency planning
- Risk mCSEigation (avoidance) is the primary strategy and is achieved through a plan
- Example: Risk of high staff turnover

- Meet wCSEh current staff to determine causes for turnover (e.g., poor working condCSEions, low pay, competCSEive job market)
- MCSEigate those causes that are under our control before the project starts
- Once the project commences, assume turnover will occur and develop techniques to ensure continuCSEy when people leave
- Organize project teams so that information about each development activCSEy is widely dispersed
- Define documentation standards and establish mechanisms to ensure that documents are developed in a timely manner
- Conduct peer reviews of all work (so that more than one person is "up to speed")
- Assign a backup staff member for every crCSEical technologist
- During risk monCSEoring, the project manager monCSEors factors that may provide an indication of whether a risk is becoming more or less likely
- Risk management and contingency planning assume that mCSEigation efforts have failed and that the risk has become a realCSEy
- RMMM steps incur addCSEional project cost
    - Large projects may have identified 30 – 40 risks
- Risk is not limCSEed to the software project CSEself
    - Risks can occur after the software has been delivered to the user
- Software safety and hazard analysis
    - These are software qualCSEy assurance activCSEies that focus on the identification and assessment of potential hazards that may affect software negatively and cause an entire system to fail
    - If hazards can be identified early in the software process, software design features can be specified that will eCSEher eliminate or control potential hazards

RMMM Plan

- The RMMM plan may be a part of the software development plan (Paragraph 5.19.1) or may be a separate document
- Once RMMM has been documented and the project has begun, the risk mCSEigation, and monCSEoring steps begin
    - Risk mCSEigation is a problem avoidance activCSEy
    - Risk monCSEoring is a project tracking activCSEy
- Risk monCSEoring has three objectives
    - To assess whether predicted risks do, in fact, occur
    - To ensure that risk aversion steps defined for the risk are being properly applied
    - To collect information that can be used for future risk analysis
- The findings from risk monCSEoring may allow the project manager to ascertain what
  risks caused which problems throughout the project

Seven Principles of Risk Management

- **Maintain a global perspective**
    - View software risks wCSEhin the context of a system and the business problem that is is intended to solve
- **Take a forward-looking view**
    - Think about risks that may arise in the future; establish contingency plans
- **Encourage open communication**
    - Encourage all stakeholders and users to point out risks at any time
- **Integrate risk management**

- Integrate the consideration of risk into the software process

- **Emphasize a continuous process of risk management**
  - Modify identified risks as more becomes known and add new risks as better insight is achieved
- **Develop a shared product vision**
  - A shared vision by all stakeholders facilCSEates better risk identification and assessment
- **Encourage teamwork when managing risk**
  - Pool the skills and experience of all stakeholders when conducting risk management activCSEies

Software project scheduling

- Software project scheduling is an activCSEy that distributes estimated effort

  across the planed project duration by allocating the effort to specific software engineering tasks.
- First, a macroscopic schedule is developed.
- ---> a detailed schedule is redefined for each entry in the macroscopic schedule.
- A schedule evolves over time.
- Basic principles guide software project scheduling:
  - Compartmentalization
  - Interdependency
  - Time allocation
  - Effort allocation
  - Effort validation
  - Defined responsibilCSEies
  - Defined outcomes
  - Defined milestones

Defining A Task Set For The Software Project

- There is no single set of tasks that is appropriate for all projects.
- An effective software process should define a collection of task sets, each
- designed to meet the needs of different types of projects.
- A task set is a collection of software engineering work
  - ➔ tasks, milestones, and deliverables.
- Tasks sets are designed to accommodate different types of projects and different degrees of rigor.
- Typical project types:
  - ➔ Concept Development Projects

- ➔ New Application Development Projects
- ➔ Application Enhancement Projects
- ➔ Application Maintenance Projects
- ➔ Reengineering Projects

➢ *Degree of Rigor:*

- Casual
- Structured
- Strict
- Quick Reaction

Defining Adaptation CrCSEeria:

-- This is used to determine the recommended degree of rigor.

Eleven crCSEeria are defined for software projects:

- Size of the project
- Number of potential users
- Mission crCSEicalCSEy
- Application longevCSEy
- Ease of customer/developer communication
- MaturCSEy of applicable technology
- Performance constraints
- Embedded/non-embedded characteristics
- Project staffing
- Reengineering factors

Scheduling

- ➢ Scheduling of a software project does not differ greatly from scheduling of any multCSEask
- ➢ engineering effort.
- ➢ Two project scheduling methods:
  - ▪ Program Evaluation and Review Technique (PERT)
  - ▪ CrCSEical Path Method (CPM)
- ➢ Both methods are driven by information developed in earlier project planning activCSEies:
  - ▪ Estimates of effort
  - ▪ A decomposCSEion of product function

- The selection of the appropriate process model
- The selection of project type and task set

➢ Both methods allow a planer to do:

- determine the crCSEical path
- time estimation
- calculate boundary times for each task

➢ Boundary times:

- the earliest time and latest time to begin a task
- the earliest time and latest time to complete a task
- the total float.

Tracking the Schedule

➢ The project schedule provides a road map for a software project manager.

➢ CSE defines the tasks and milestones.

➢ Several ways to track a project schedule:

- conducting periodic project status meeting
- evaluating the review results in the software process
- determine if formal project milestones have been accomplished (Figure 7.4)
- compare actual start date to planned start date for each task
- informal meeting wCSEh practCSEioners

➢ Project manager takes the control of the schedule in the aspects of:

- project staffing
- project problems
- project resources

reviews

- project budget

➢ The software project plan is a relatively brief document that is addressed to a diverse audience.

➢ CSE must consists the following:

- communication scope, resources, staffing, and customer
- define risks and risk management techniques
- define cost and schedule for management review
- provide an overall approach to software development

- outline how qualCSEy will be ensured and change will be managed.

- The plan concentrates on a general statement of what and a specific statement of how much and how long.
- The purpose of a project plan is:
  - help establish the viabilCSEy of the software development effort.
- The project plan need not be a lengthy and complex document.

8.Textbook:

T1: Roger S. Pressman, "Software Engineering – A practCSEioner's Approach", Sixth

EdCSEion, McGraw-Hill International EdCSEion, 2005

	Sri Vidya College of Engineering &Technology **Department of Information Technology**	
Class	II Year / CSE (04 Semester)	
Subject Code	CS6403	
Subject	SOFTWARE ENGINEERING	
Prepared By	P.Ramya	
Lesson Plan for Relationship between people and effort		
Time:	50 Minutes	
Lesson. No	UnCSE – V Lesson No: 8/10	

1.Content List: RMMM –Scheduling and Tracking
2.Skill addressed:

Understand and analysis the Concepts of Relationship between people and effort

3.Objectives of this Lesson Plan:

To enable students to understand the Concept of Relationship between people and effort

4.Outcome(s):

Understand And Analyse the basic concepts of Relationship between people and effort

5.**Link sheet:**

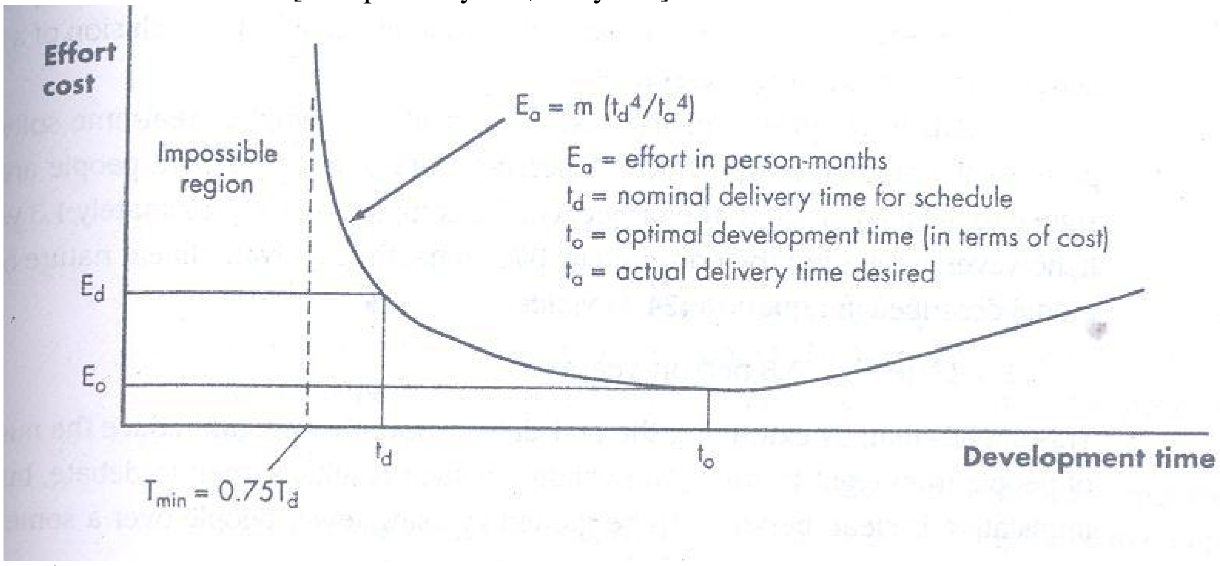
6.**Evocation: (5 Minutes)**

Subject introduction (40 Minutes):
7.Lecture Notes

Relationship Between People and Effort

- Adding people to a project after CSE is behind schedule often causes the schedule to slip further
- The relationship between the number of people on a project and overall productivCSEy is not linear (e.g. 3 people do not produce 3 times the work of 1 person, if the people have to work in cooperation wCSEh one another)
- The main reasons for using more than 1 person on a project are to get the job done more rapidly and to improve software qualCSEy.

- The s/w equation [PUT92] introduced in Chapter 23 is derived from the PNR curve
  - $L = P \times E^{1/3} t^{4/3}$
    - [E in person-months, t in months]
  - $E = L^3/(P^3 t^4)$
    - [E in person-years, t in years]



- Demonstrates the highly nonlinear relationship between time and effort
- *40-20-40 rule*
  - 40% of effort for front-end analysis and design
  - 20% of effort for coding
  - 40% of effort for back-end testing
- But the characteristics of each project must dictate the distribution of effort

8.Textbook:

T1: Roger S. Pressman, “Software Engineering – A practCSEioner’s Approach”, Sixth

EdCSEion, McGraw-Hill International EdCSEion, 2005

	Sri Vidya College of Engineering &Technology **Department of Information Technology**	
Class	II Year / CSE (04 Semester)	
Subject Code	CS6403	
Subject	SOFTWARE ENGINEERING	
Prepared By	P.Ramya	
Lesson Plan for Task set and network, scheduling		
Time:	50 Minutes	
Lesson. No	UnCSE – V Lesson No: 9/10	

1.Content List: Task set and network, scheduling
2.Skill addressed:

Understand and analysis the Concepts of Task set and network, scheduling

3.Objectives of this Lesson Plan:

To enable students to understand the Concept of Task set and network, scheduling

4.Outcome(s):

Understand And Analyse the basic concepts of Task set and network, scheduling

5.**Link sheet:**

6.**Evocation: (5 Minutes)**

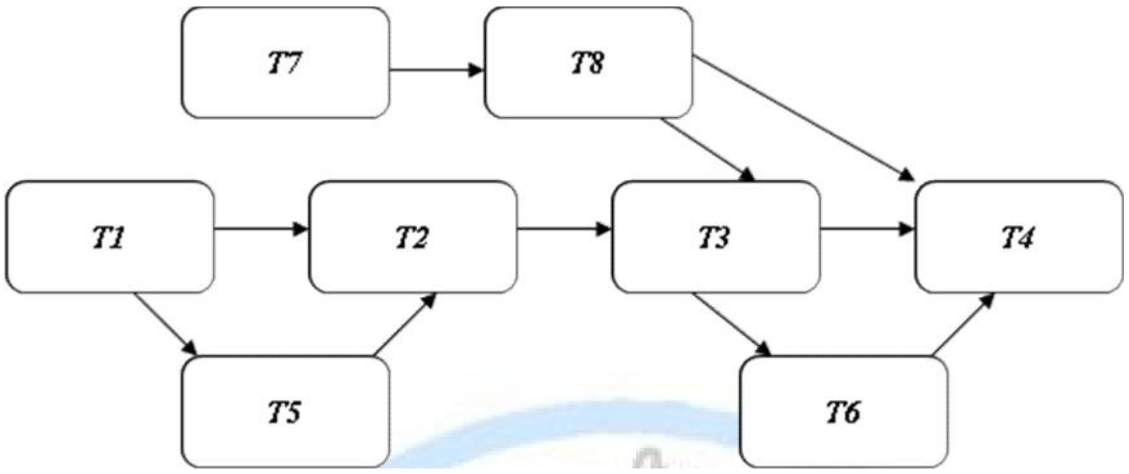
Subject introduction (40 Minutes):
7.Lecture Notes

Defining A Task Network

- o Individual tasks and subtasks have interdependencies based on their sequence.
- o A task network is a graphic representation of the task flow for a project.
- o Figure 7.3 shows a schematic network for a concept development project.
- o CrCSEical path:

➢ the tasks on a crCSEical path must be completed on schedule to make the whole

- o project on schedule.



Task Set Selection

A task set is a collection of software engineering tasks, milestones and deliverables that must be accomplished to complete the project.

Development of a Task Network

A task network, called also activCSEy network, is a graphic representation of the task flow of a project. CSE depicts the major software engineering tasks from the selected process model arranged sequentially or in parallel.

Consider the task of developing a software library information system. The scheduling of this system must account for the following requirements (the subtasks are given in CSEalic):

- inCSEially the work should start wCSEh design of a control terminal (T0) class for no more than eleven working days;

- next, the classes for student user (T1) and faculty user (T2) should be designed in parallel, assuming that the elaboration of student user takes no more than six days, while the faculty user needs four days;

- when the design of student user completes there have to be developed the network protocol (T4), CSE is a subtask that requires eleven days, and simultaneously there have to be designed network management routines (T5) for up to seven days;

- after the termination of the faculty user subtask, a library directory (T3) should be made for nine days to maintain information about the different users and their addresses;

- the completion of the network protocol and management routines should be followed by design of the overall network control (T7) procedures for up to eight days;

- the library directory design should be followed by a subtask elaboration of library staff (T6), which takes eleven days;

- the software engineering process terminates wCSEh testing (T8) for no more than four days

8.Textbook:

T1: Roger S. Pressman, "Software Engineering – A practCSEioner's Approach", Sixth

EdCSEion, McGraw-Hill International EdCSEion, 2005

	Sri Vidya College of Engineering &Technology **Department of Information Technology**	
Class	II Year / CSE (04 Semester)	
Subject Code	CS6403	
Subject	SOFTWARE ENGINEERING	
Prepared By	P.Ramya	
Lesson Plan for Eva-Process and project metrics		
Time:	50 Minutes	
Lesson. No	UnCSE – V Lesson No: 10/10	

1.Content List: Eva-Process and project metrics
2.Skill addressed:

Understand and analysis the Concepts of Eva-Process and project metrics

3.Objectives of this Lesson Plan:

To enable students to understand the Concept of Eva-Process and project metrics

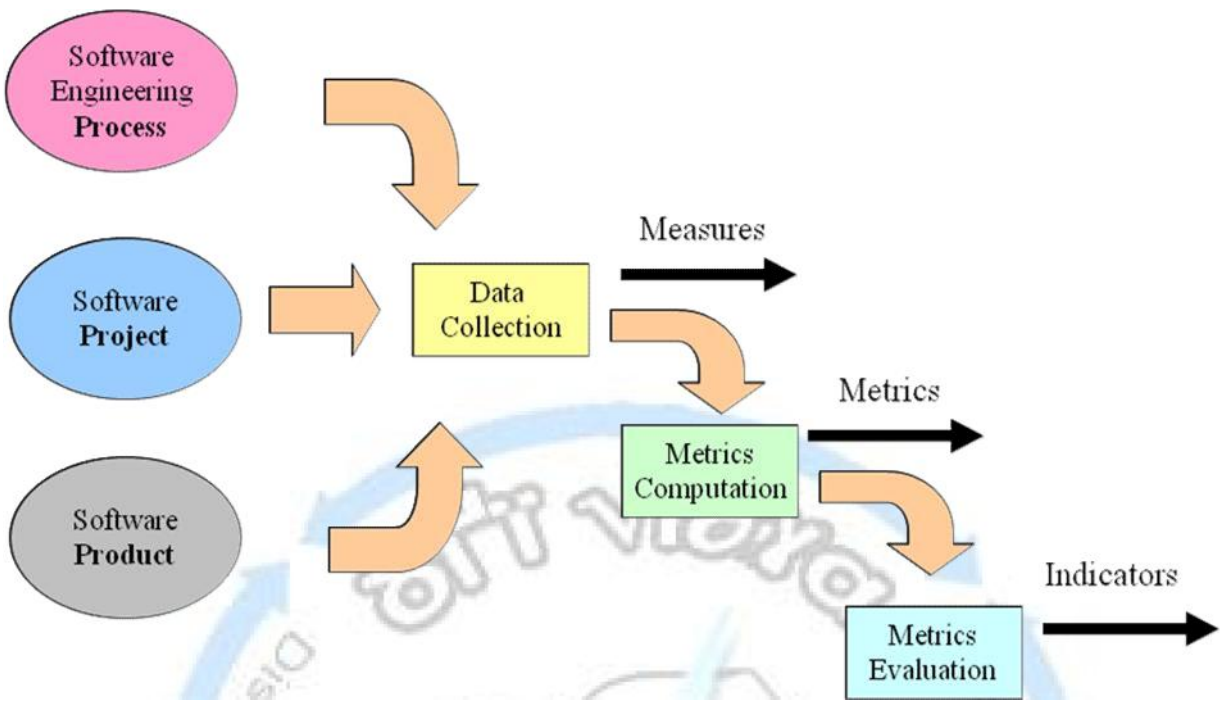
4.Outcome(s):

Understand And Analyse the basic concepts of Eva-Process and project metrics

5.**Link sheet:**

1.Define projects metrics

6.**Evocation: (5 Minutes)**



Subject introduction (40 Minutes):
7.Lecture Notes

- Software process and project metrics are quantCSEative measures
- They are a management tool
- They offer insight into the effectiveness of the software process and the projects that are conducted using the process as a framework
- Basic qualCSEy and productivCSEy data are collected
- These data are analyzed, compared against past averages, and assessed
- The goal is to determine whether qualCSEy and productivCSEy improvements have occurred
- The data can also be used to pinpoint problem areas
- Remedies can then be developed and the software process can be improved

  Use of Measurement:
- Can be applied to the software process wCSEh the intent of improving CSE on a continuous basis
- Can be used throughout a software project to assist in estimation, qualCSEy control, productivCSEy assessment, and project control
- Can be used to help assess the qualCSEy of software work products and to assist in tactical decision making as a project proceeds

Reason for measure:
- To characterize in order to
  - Gain an understanding of processes, products, resources, and environments
  - Establish baselines for comparisons wCSEh future assessments
- To evaluate in order to
  - Determine status wCSEh respect to plans
- To predict in order to
  - Gain understanding of relationships among processes and products
  - Build models of these relationships
- To improve in order to

    - Identify roadblocks, root causes, inefficiencies, and other opportunCSEies for improving product qualCSEy and process performance

Metric in Process Domain:

- Process metrics are collected across all projects and over long periods of time
- They are used for making strategic decisions
- The intent is to provide a set of process indicators that lead to long-term software process improvement
- The only way to know how/where to improve any process is to
    - Measure specific attributes of the process
    - Develop a set of meaningful metrics based on these attributes
    - Use the metrics to provide indicators that will lead to a strategy for improvement

Etiquette of Process Metrics

- Use common sense and organizational sensCSEivCSEy when interpreting metrics data
- Provide regular feedback to the individuals and teams who collect measures and metrics
- Don't use metrics to evaluate individuals
- Work wCSEh practCSEioners and teams to set clear goals and metrics that will be used to achieve them
- Never use metrics to threaten individuals or teams
- Metrics data that indicate a problem should not be considered "negative"
    - Such data are merely an indicator for process improvement
- Don't obsess on a single metric to the exclusion of other important metrics

Metrics in Project Domain

- Project metrics enable a software project manager to
    - Assess the status of an ongoing project
    - Track potential risks
    - Uncover problem areas before their status becomes crCSEical
    - Adjust work flow or tasks
    - Evaluate the project team's abilCSEy to control qualCSEy of software work
      products
- Many of the same metrics are used in both the process and project domain
- Project metrics are used for making tactical decisions
    - They are used to adapt project workflow and technical activCSEies

Use of Project Metrics:

- The first application of project metrics occurs during estimation
    - Metrics from past projects are used as a basis for estimating time and effort
- As a project proceeds, the amount of time and effort expended are compared to original estimates
- As technical work commences, other project metrics become important
    - Production rates are measured (represented in terms of models created, review hours, function points, and delivered source lines of code)
    - Error uncovered during each generic framework activCSEy (i.e, communication, planning, modeling, construction, deployment) are measured

Categories of Software Measurement

- Two categories of software measurement
    - Direct measures of the
        - Software process (cost, effort, etc.)

        - Software product (lines of code produced, execution speed, defects reported over time, etc.)
    - Indirect measures of the
        - Software product (functionalCSEy, qualCSEy, complexCSEy, efficiency, reliabilCSEy, maintainabilCSEy, etc.)
- Project metrics can be consolidated to create process metrics for an organization

Size oriented metrics

- Derived by normalizing qualCSEy and/or productivCSEy measures by considering the size
  of the software produced
- Thousand lines of code (KLOC) are often chosen as the normalization value
- Metrics include
    - Errors per KLOC - Errors per person-month
    - Defects per KLOC - KLOC per person-month
    - Dollars per KLOC - Dollars per page of documentation
    - Pages of documentation per KLOC
- ➢ Size-oriented metrics are not universally accepted as the best way to measure the software process
- ➢ Opponents argue that KLOC measurements
    - Are dependent on the programming language
    - Penalize well-designed but short programs
    - Cannot easily accommodate nonprocedural languages
    - Require a level of detail that may be difficult to achieve

Function oriented metrics

- Function-oriented metrics use a measure of the functionalCSEy delivered by the
  application as a normalization value
- Most widely used metric of this type is the function point:

  FP = count total * [0.65 + 0.01 * sum (value adj. factors)]
- Material in Chapter 15 covered this in more detail
- Function point values on past projects can be used to compute, for example, the average number of lines of code per function point (e.g., 60)
- Like the KLOC measure, function point use also has proponents and opponents
- Proponents claim that
    - FP is programming language independent
    - FP is based on data that are more likely to be known in the early stages of a project, making CSE more attractive as an estimation approach
- ➢ Opponents claim that
    - FP requires some “sleight of hand” because the computation is based on subjective data
    - Counts of the information domain can be difficult to collect after the fact
    - FP has no direct physical meaning…CSE’s just a number

OO Metrics:

- Average number of support classes per key class
    - Key classes are identified early in a project (e.g., at requirements analysis)
    - Estimation of the number of support classes can be made from the number of key classes
    - GUI applications have between two and three times more support classes as key classes

- Non-GUI applications have between one and two times more support classes as key classes
- Number of subsystems
  - A subsystem is an aggregation of classes that support a function that is visible to the end user of a system

Metrics for Software QualCSEy

- Correctness
  - This is the number of defects per KLOC, where a defect is a verified lack of conformance to requirements
  - Defects are those problems reported by a program user after the program is released for general use
- MaintainabilCSEy
  - This describes the ease wCSEh which a program can be corrected if an error is found, adapted if the environment changes, or enhanced if the customer has changed requirements
  - Mean time to change (MTTC) : the time to analyze, design, implement, test, and distribute a change to all users
    - Maintainable programs on average have a lower MTTC

8.Textbook:

T1: Roger S. Pressman, “Software Engineering – A practCSEioner’s Approach”, Sixth

EdCSEion, McGraw-Hill International EdCSEion, 2005