UNIVERSITY of HOUSTON

DEPARTMENT OF COMPUTER SCIENCE

COSC 4351 Fall 2023

Software Engineering

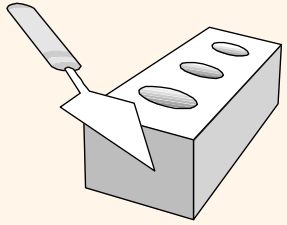
M & W 2:30 to 4 PM

Prof. Victoria Hilford

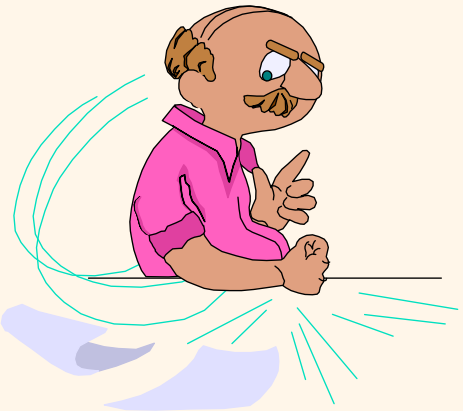
PLEASE TURN your webcam ON

NO CHATTING during LECTURE

COSC 4351
2:30 to 4



PLEASE
LOG IN
CANVAS

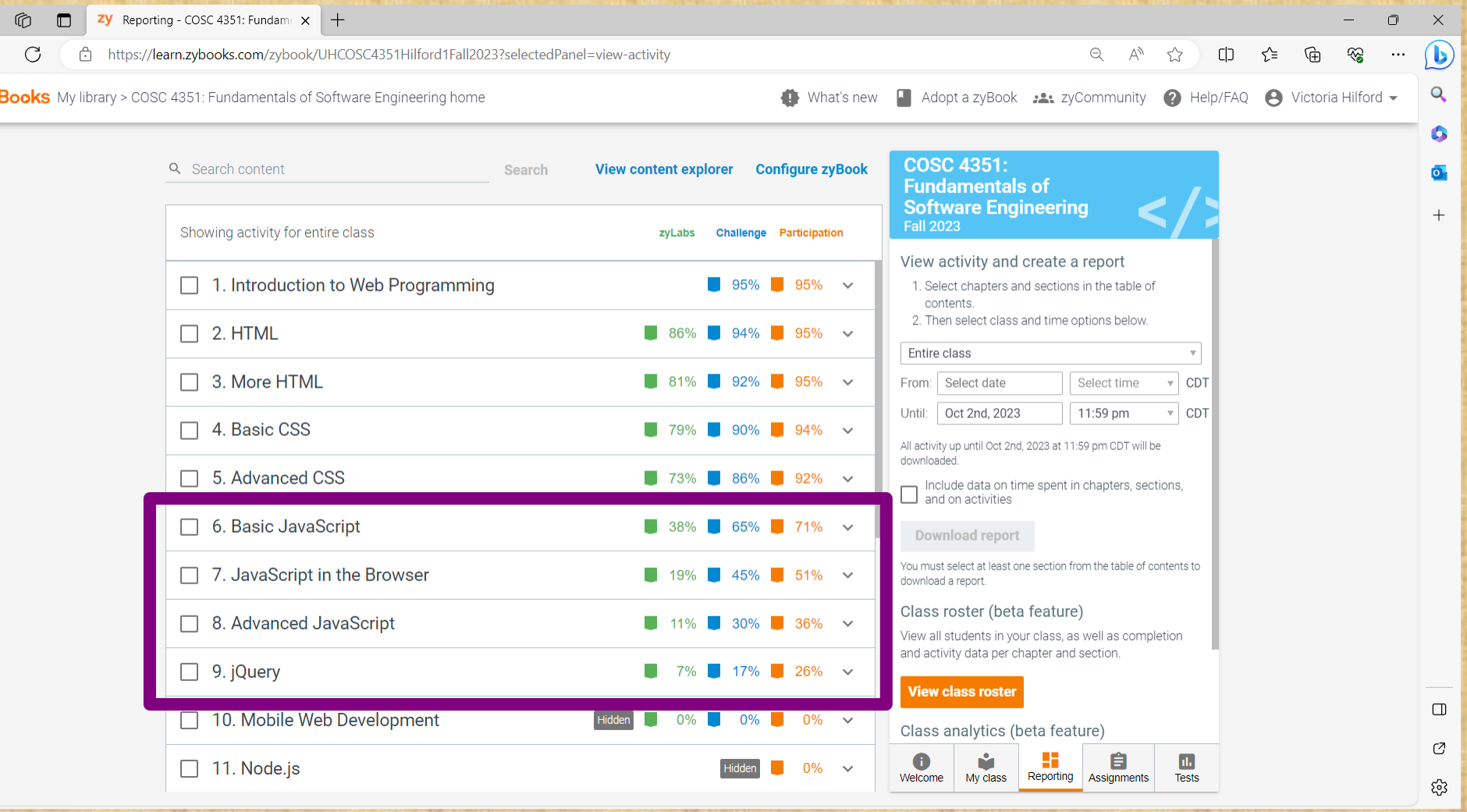


Ethan [A-L]

Jordan [M-Z]

Please close all other windows.

As of 10/02/2023 1:00 PM



On OO Languages Assignment

OO Languages

Published | Edit

Complete your UML Model MVC Class Diagram to have the pseudocode for the methods you plan to implement (minimum restoreCatalog, saveCatalog, 1. addDVD, 4. display by Category).

MUST attach it to this assignment.

TA, please do not grade unless updated UML CLASS DIAGRAM is attached.

Yours
1. Textual Analysis
2. Copy, Paste and Rearrange

Please follow the instructions in the .doc for things that need to be turned in.

Please name the projects:

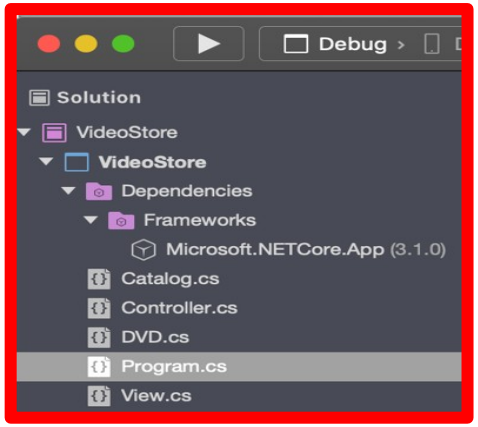
DVDApplicationCSharp.zip

DVDApplicationRuby.zip

DVDApplicationPython.zip

DVDApplicationPhp.zip

Attach these zips to this assignment.



Each unit of work should be delegated to its own unit testable descriptively named method. *Do this and all your methods end up tiny and readable without ever counting lines.....*

24 Lines of Code

Complete your UML Model MVC Class Diagram to have the pseudocode for the methods you plan to implement

Below is a list of the lines of code of some of the most widely used operating systems on the market:

- MS-DOS – 4000 lines of code
- Windows NT – 2 million lines of code
- Windows 3.1 – 2 million lines of code
- Windows 7 – 40 million lines of code
- Windows XP – 45 million lines of code

10.02.2023 (M 2:30 to 4) (12)		Lecture 4: Estimating and Planning Tutorials 4 COCOMO and MS Project		Estimating and Planning Papers Summary (1 Page) CANVAS Assignment	
10.04.2023 (W 2:30 to 4) (13)		EXAM 2 REVIEW (CANVAS) (ZyBook)	Download ZyBook: Sections 6-9		
10.09.2023 (M 2:30 to 4) Optional (14)					Q & A Set 2 topics.
10.11.2023 (W 2:30 to 4) (15)					EXAM 2 (CANVAS) (ZyBook)

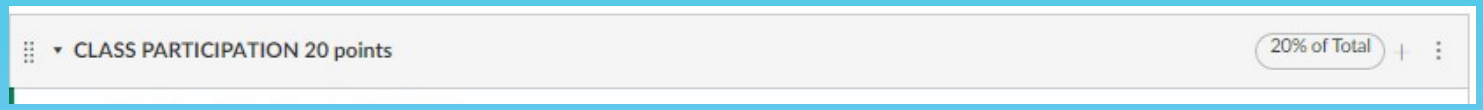
Class 12

Lecture 4
on estimating and planning

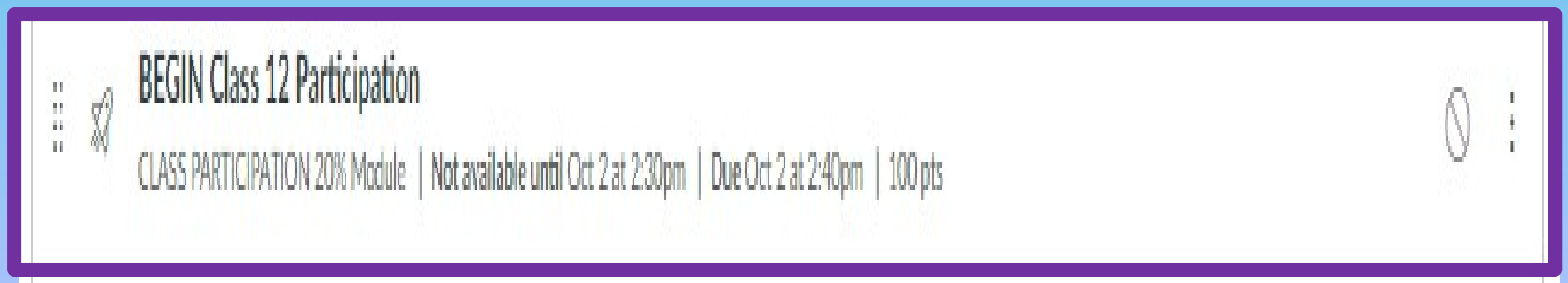
10.02.2023 (M 2:30 to 4) (12)		Lecture 4: Estimating and Planning Tutorials 4 COCOMO and MS Project		Estimating and Planning Papers Summary (1 Page) CANVAS Assignment	

From 2:30 to 2:40 – 10 minutes.

10.02.2023 (M 2:30 to 4) (12)		Lecture 4: Estimating and Planning Tutorials 4 COCOMO and MS Project		Estimating and Planning Papers Summary (1 Page) CANVAS Assignment	



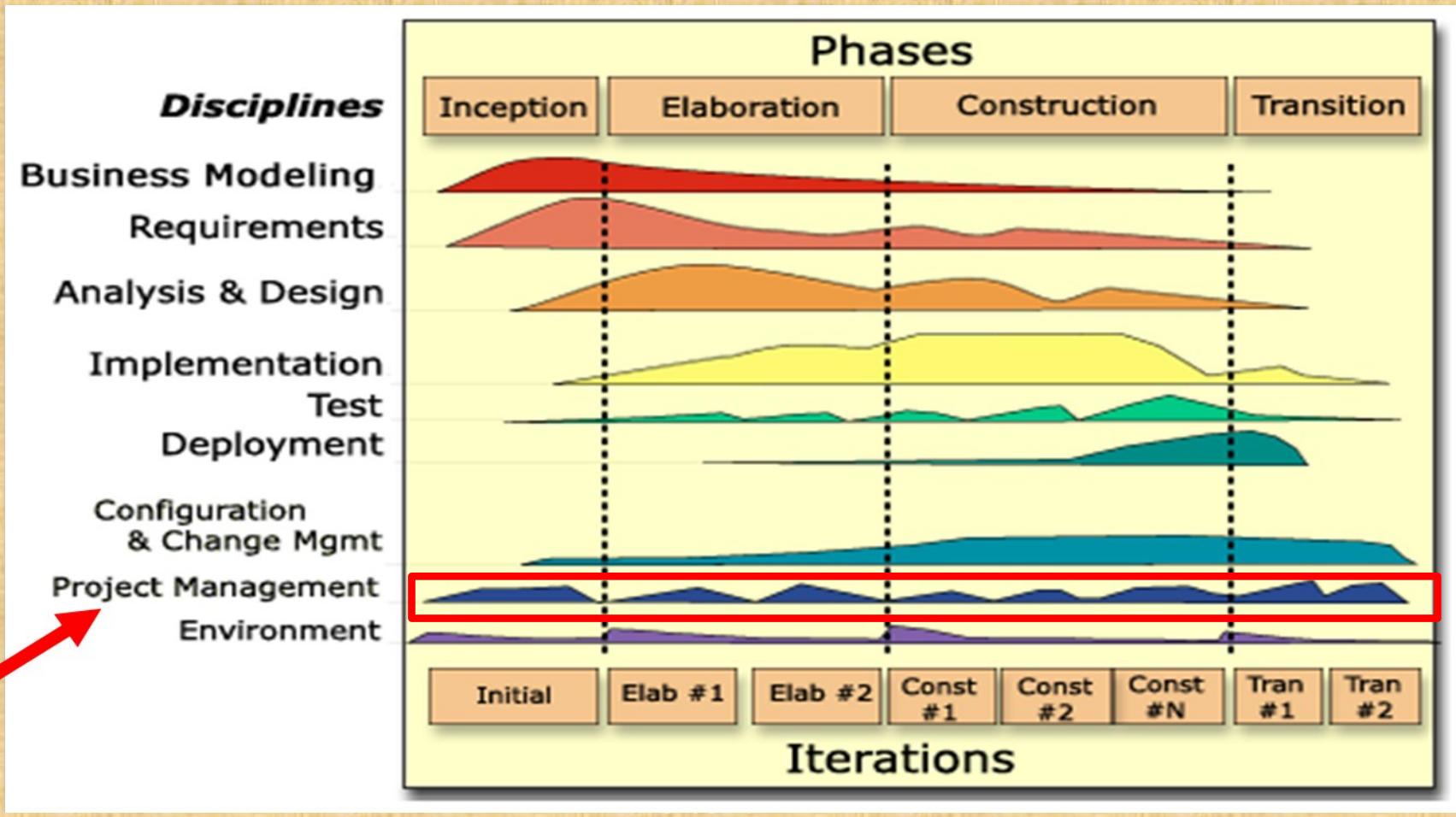
PASSWORD: CLASS 12



VH– Publish.

From 2:40 to 3:30 – 50 minutes.

Planning and Estimating



Planning and Estimating

- Before starting a **software project**, it is essential **to Plan** the entire operation in detail.
- Once the **project** begins, management (TLs) **must closely monitor progress**, noting deviations from **the Plan** and taking corrective action where necessary.
- Also, it is vital that the **Client** be provided accurate Estimates of **how long the project** will take and **how much it** will **cost**.
- Different estimation techniques are presented, including **Function Points** and **COCOMO II**.

2002 survey of information technology organizations
- 78% have been involved in disputes ending in litigation

For the organizations that entered into litigation:
- In 67% of the disputes, the functionality of the information system as delivered did not meet up to the claims of the developers
- In 56% of the disputes, the promised delivery date slipped several times
- In 45% of the disputes, the defects were so severe that the information system was unusable

Objectives

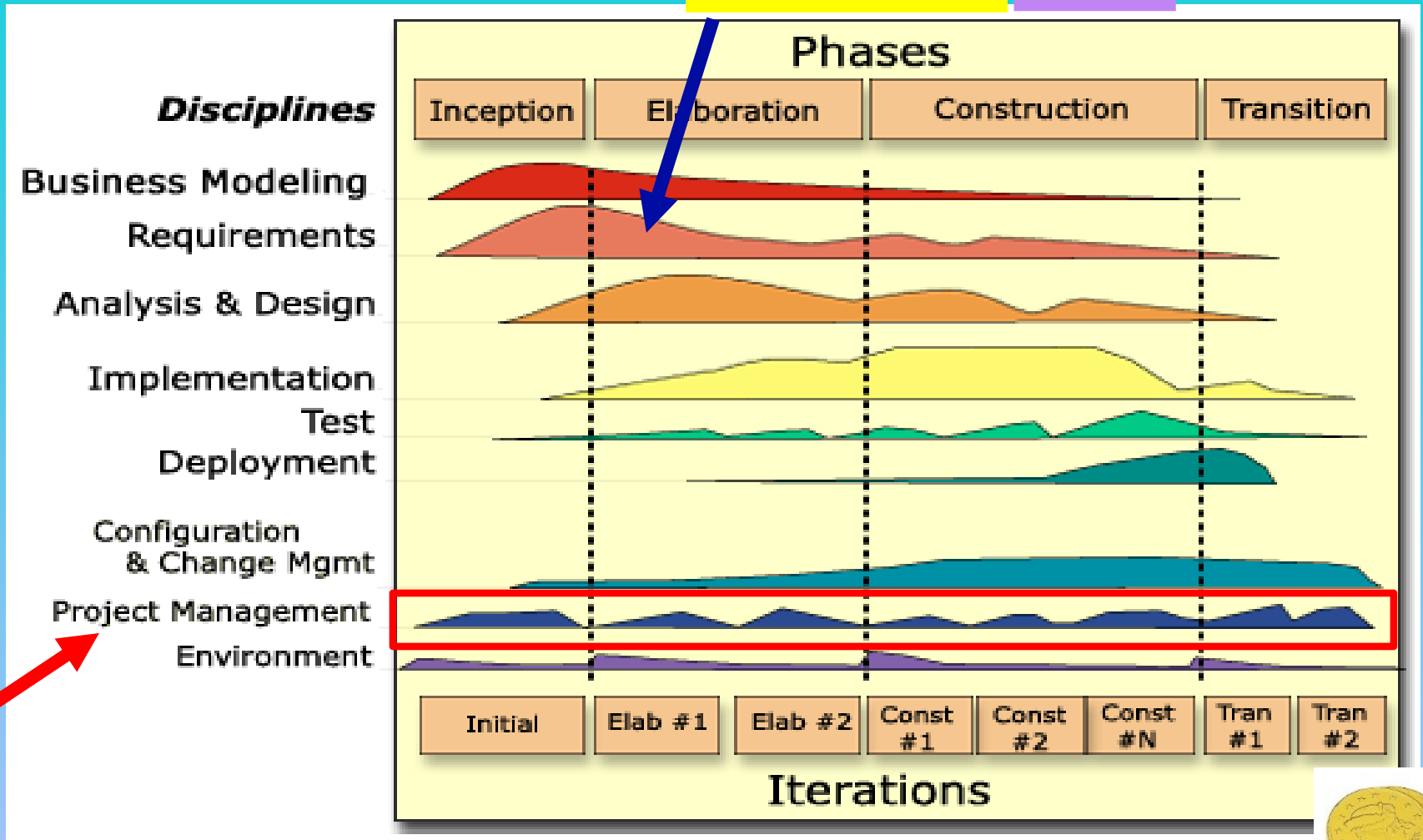
The importance of **Planning**

Estimate the **Size** and **Cost** of building a software product

Appreciate the importance of **updating and tracking Estimates**

Unified Process

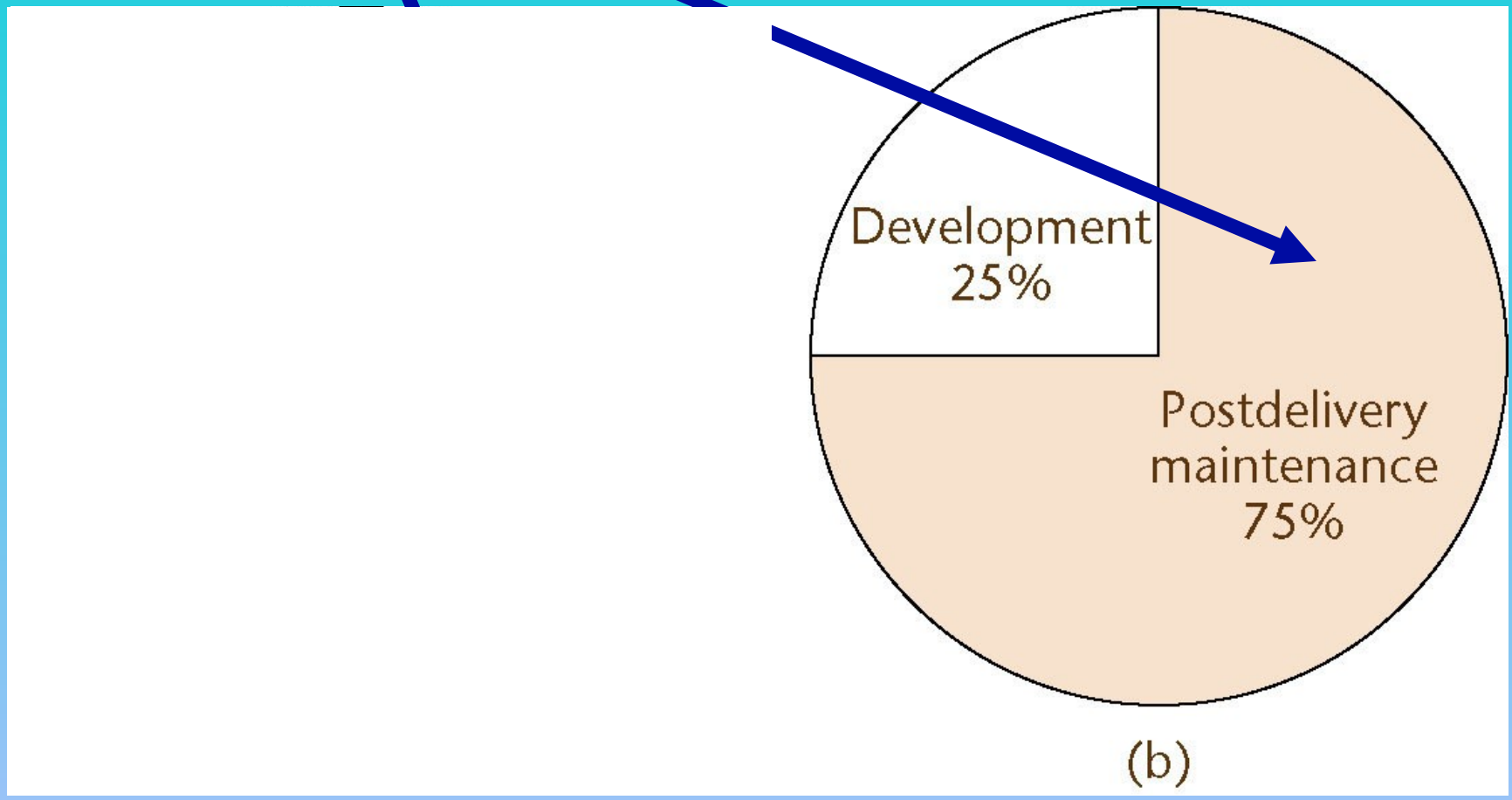
SRS & DRS ????



SRS - Software Requirements Specification

DRS – Data Requirements Specification

Time (= Cost) of Postdelivery Maintenance



(a) Between 1976 and 1981

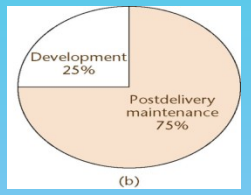
(b) Between 1992 and 1998

Planning and Estimating

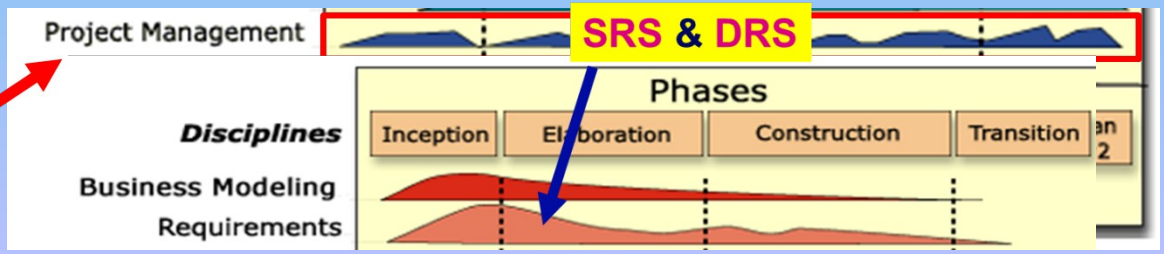
Before starting to build **software**, *it is essential* to **Plan** the *entire* development **effort** *in detail*

For large software projects, failure is generally determined early in the process, because failures almost exclusively have to do with planning: the failure to create a workable plan to stick to it, or both. Healthcare.gov reportedly involved over fifty-five contractors, managed by a human-services agency that lacked deep experience in software engineering or project management.

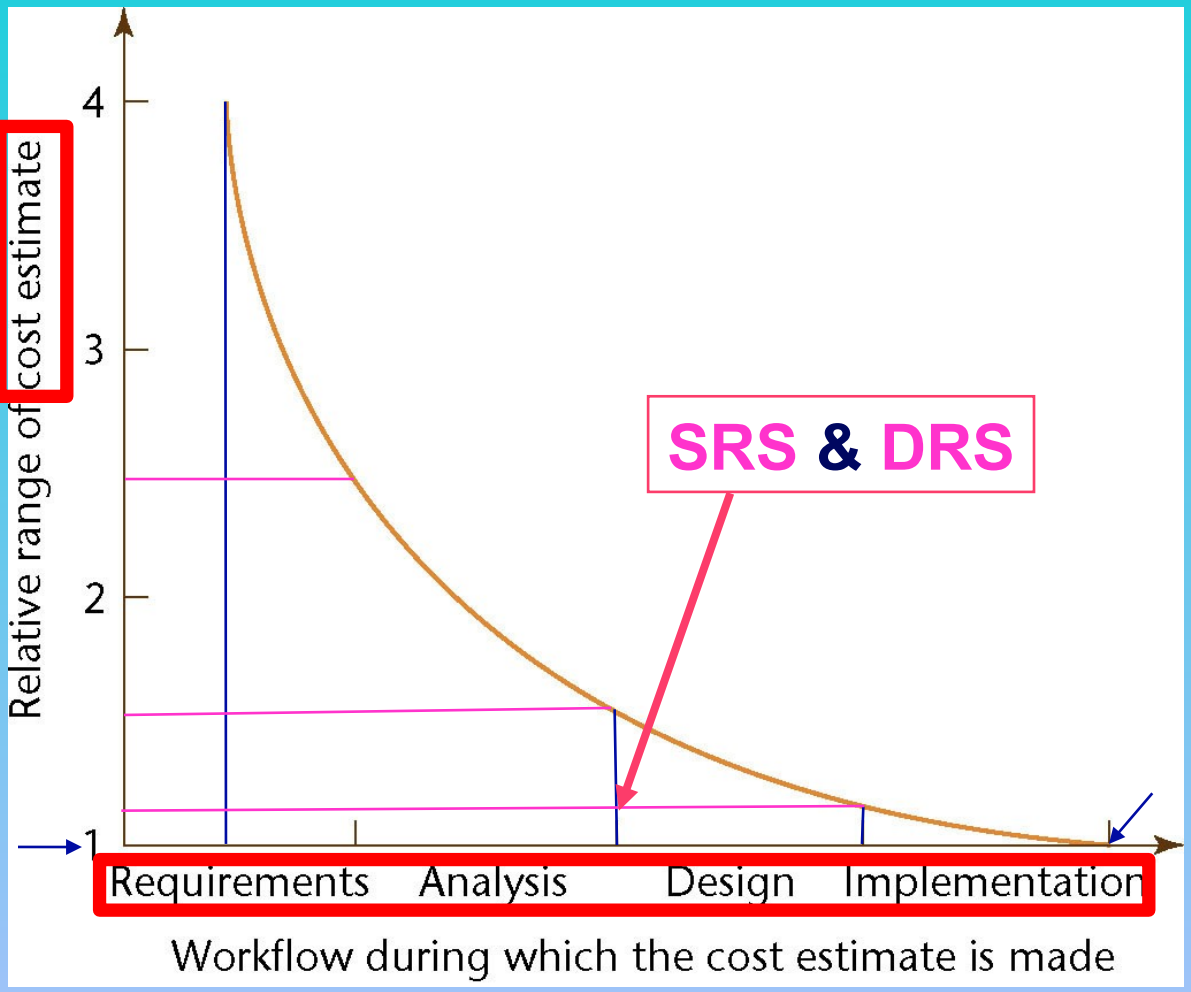
Planning continues during ***Development*** and then ***Postdelivery Maintenance***



- Initial **Planning** is not enough
- **Planning** must proceed throughout **the project**
- The **earliest possible time that detailed Planning** can take place is *after the **specifications**/**analysis** are complete (**SRS**/**DRS**)*



Planning and the Software Process

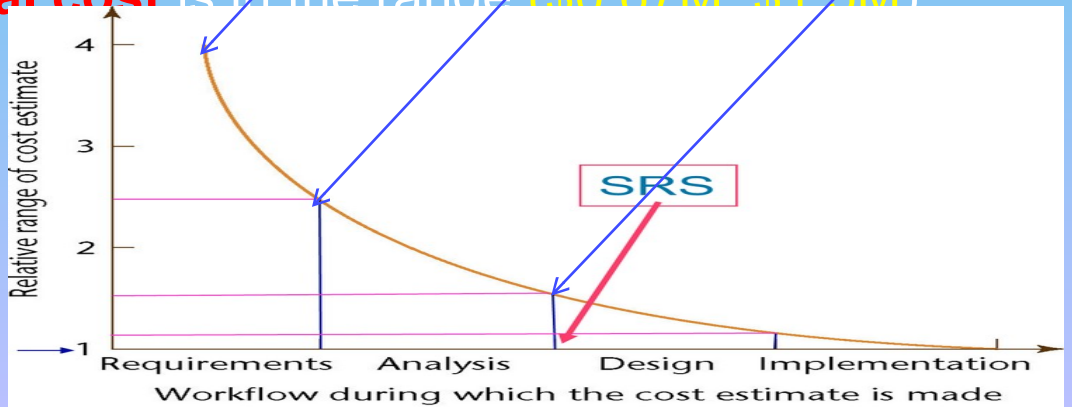


The accuracy of Estimation increases as the **process** proceeds

Planning and the Software Process

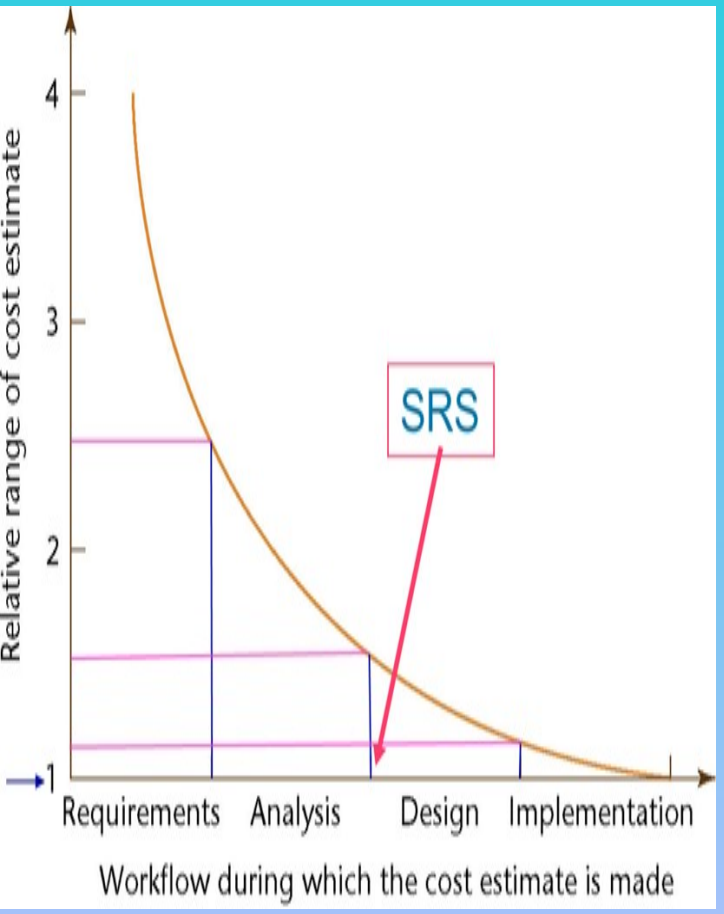
Example

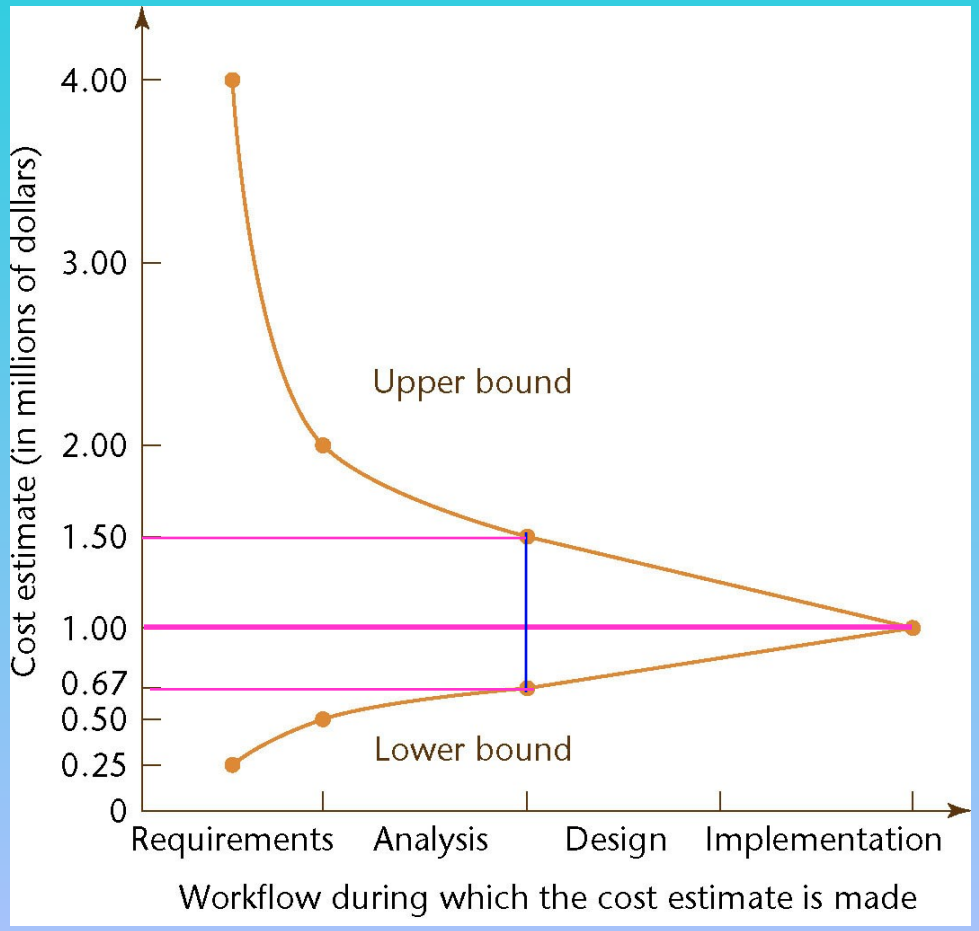
- **Cost estimate** of $1 million *during the Requirements Workflow*
  - Likely **actual cost** is in the range ($0.25M, $4M)
- **Cost estimate** of $1 million at end of the *Requirements Workflow*
  - Likely **actual cost** is in the range ($0.5M, $2.5M)
- **Cost estimate** of $1 million at the end of the *Analysis Workflow* (**SRS** & **DRS** **earliest appropriate time**)
  - Likely **actual cost** is in the range ($0.67M, $1.5M)



Planning and the Software Process

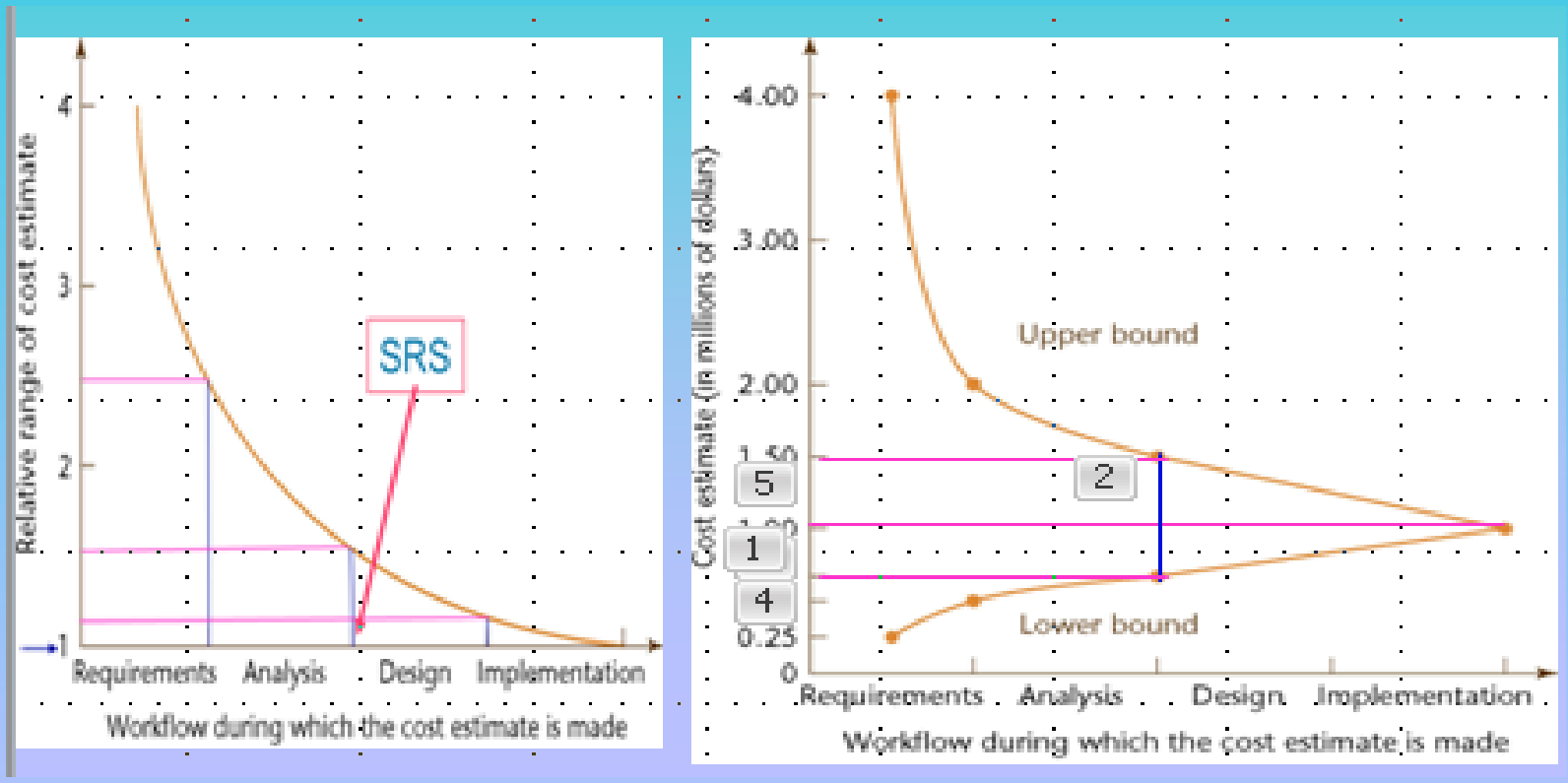
- These points are shown in the ***cone of uncertainty***





Planning and the Software Process

- This model is old (1976)
  - **Estimating** techniques **have improved**
  - But the **shape of the curve is likely to be similar**



Estimating Duration and Cost

- **Accurate Cost *Estimation*** is **critical**
  - Internal (**Cost** to **Developers**), external **Costs** (**price** that the **Client** pays)
- **Accurate Duration *Estimation*** is **critical**

SRS & DRS

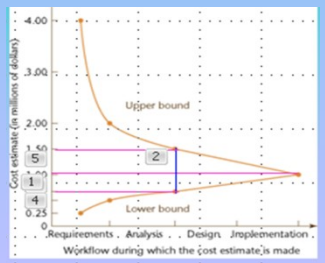
- There are too many variables for **accurate Estimate** of **Cost** or **Duration**

2002 survey of information technology organizations
- 78% have been involved in disputes ending in litigation

For the organizations that entered into litigation:
- In 67% of the disputes, the functionality of the information system as delivered did not meet up to the claims of the developers
- In 56% of the disputes, the promised delivery date slipped several times
- In 45% of the disputes, the defects were so severe that the information system was unusable

Estimates made *before the specifications* have been **signed off** by the **Client** are *less acurate* than an **Estimate** made when sufficient data have accumulated (**SRS & DRS Contracts**).

Estimate vs. Actual



Impediment to Accurate Estimates: Human Factors

Sackman (**1968**) measured differences of up to 28 to 1 between pairs of programmers (*10 years* to beginners)

- He compared matched pairs of programmers with respect to
  - **Product** size 6 to 1
  - **Product** execution time 8 to 1
  - **Development** time 9 to 1
  - **Coding** time 18 to 1
  - **Debugging** time 28 to 1

Critical **staff members** may resign during the **project** (*Time and Money* ***are spent*** attempting to fill the vacated position)

Metrics for the Size of a Product

The most common **Metric** for the *size of a* ***Product*** is the number of Lines Of Code

- Lines Of Code (LOC), Thousand Delivered Source Instructions (KDSI), Thousand Lines Of Code (KLOC)

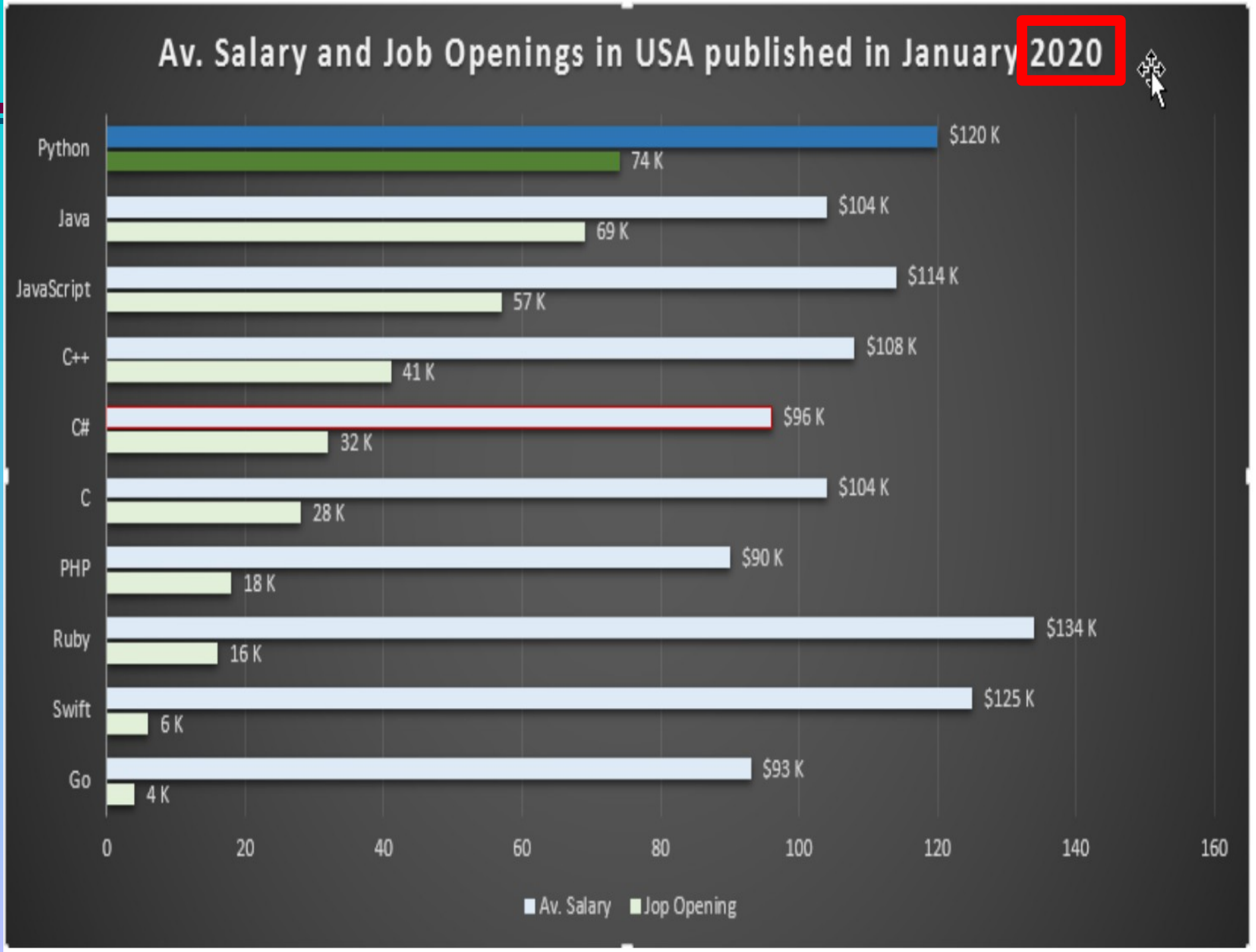
Windows XP – 45 million lines of code

There are many problems associated with the **use of Lines Of Code** (LOC):

- Source code is only a small part of the total **software** Development effort
- Source code implemented in two different languages different # **LOC**
- How to count LOC?
- Not all code is delivered to the **Client** (tools used during **Development**)
- Uses code, report, screen, **GUI** generators

- ***Actual # LOC*** can be calculated **only when** the **Product** is finished!

Estimate vs. Actual


Av. Salary and Job Openings in USA published in January 2020
Python
$120 K
74 K
Java
$104 K
69 K
JavaScript
$114 K
57 K
C++
$108 K
41 K
C#
$96 K
32 K
C
$104 K
28 K
PHP
$90 K
18 K
Ruby
$134 K
16 K
Swift
$125 K
6 K
Go
$93 K
4 K
0
20
40
60
80
100
120
140
160
Av. Salary
Jop Opening

Estimating Exercise

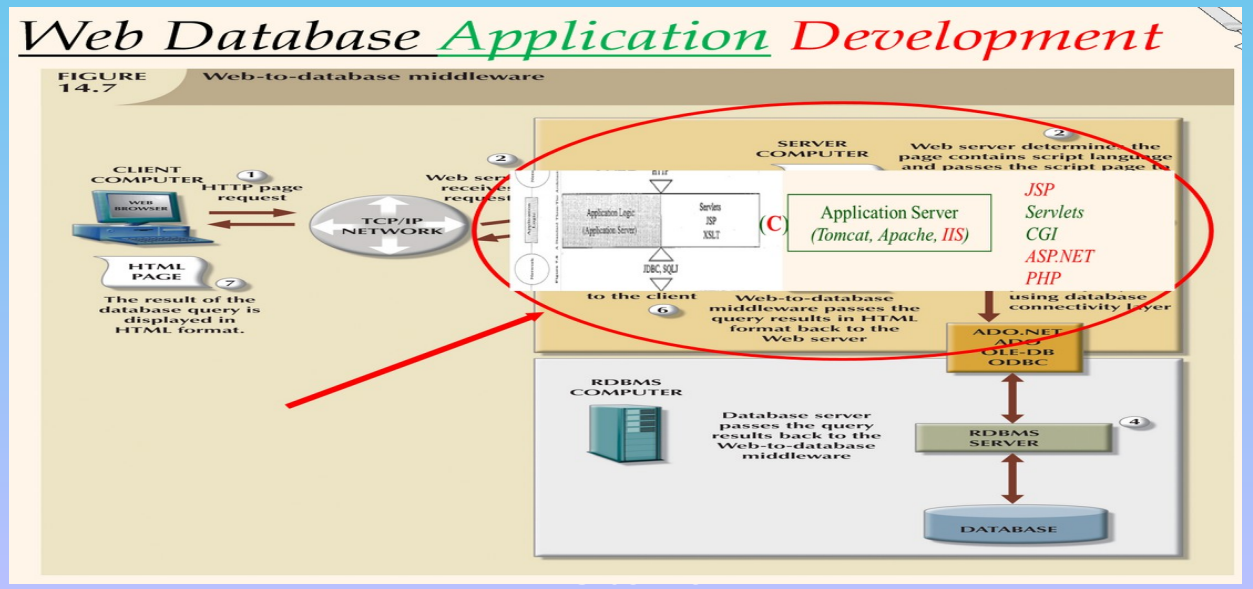
Attached Files: Estimate DVDCollection Web Application.docx (23.502 KB)

Download the attached word document.

Estimate the LOC, DURATION (in days) and COST (in dollars) for the DVD Collection Web Application.

Upload the completed word document.

You have 5 minutes to complete.



Estimate DVDCollection Web Application

DVD Collection Application Requirements:

A video store wishes to automate their processes related to their collection of DVDs. The data for each DVD will consist of title, a category, running time, year of release, and price. These DVDs are stored in Catalog.txt.

The video store processes are:

1. add new DVDs to the collection,
2. remove a DVD,
3. edit the information stored for a DVD,
4. display all DVDs by a specified category,
5. display the collection of DVDs sorted by year

The client is estimating that there will not own more than 100 DVDs.

Catalog.txt

Amadeus,Drama,160 Mins.,1984, 14.83

As Good As I

COSC 4351

Estimating Exercise

Full Stack Web Applications Java, C#, Ruby, Python, or Php5.

For the DVD Collection Application below create a website in **Java**, **C#**, **Ruby**, **Python**, or **Php5** for front end and a database backend as **MS SQLSERVER**, **ORACLE**, or **MySQL** database management systems (DBMS).

Estimate the LOC (lines of code that you will write): _____ LOC.

Estimate duration (in days): _____ days.

Estimate cost (in dollars): _____ dollars.

5 minutes

VH, Estimating DVD Web APP 2.27.2023 JUST DATA.xlsx

LOC	DURATION (days)	DVD WEB APP EST COST
10,000	14	$1,000
2,000	21	$1,000
520	28	$300
1,000	5	$500
1,500	32	$5,000
500	30	$250
1,500	14	$250
400	4	4 * 8 * 56.5
		$1,808
250	1	
500	3	$450
4,000	14	$12,000
1,000	7	$2,520
1,000	15	$500

200	8	$100,000
700	60	$1,000
500	20	$300
3,000	20	$8,000
600	14	$2,500
2,500	45	$4,000
5,000	30	$15,000
750	45	$2,000
500	10	$250
500	3	$100
3,000-4,000	20	$1,000
1,000	7	$2,000
600	30	$2,500
1,000	30	$0
15,000	60	$10,000

2,000	60	$10,000
200	30	$300
800	10	$400
500	10	$500
1,000	20	$500
1,350	2	$60
90	30	$80,000
300	4	$100
5,000	30	$15,000
250	15	$3,000
1,500	5	$1,500
900	6	$500
1,500	45	$15,000
2,000		
850		
750		
3,000	30	$5,000
5,000	40	$300
50	10	$500
5,000	20	$10,000

850	35	$2,000
1,800	30	$300
1,500	50	$8,000
500	5	$15
1,200-1,500	30	$4,500
500-1,000	5	$10,000
5,000	31	$3,000
10,000	100	$100,000
100	10	$1,000
1,000	70	$9,000
2,000	30	$50,000
2,000	14	$10
500	30	$1,000
1,250-1,750	7	450-650
2,000	60	$500
1,000	14	$500

10,000	30	$250
1,200	4	$2,000
1,000,000	60	$2,000,000
350	4	$800
15,000	30	$3,000
90	30	$80,000
500-600		$1,000-$1,500
700-900	15	$2,00-$2,500
500	14	$2,000
750-1,000	15	$2,000
		$3,000
		$100
		$500
		$2,000
		$36,000 - $56,000
		$12,000-$20,000
		$300
1,500	14	$2,000
6,000	20	$10,000
300	3	$150

DVDCollection Web App – 50 – 1,000,000 LOC (min & max)

10 – 2,000,000 $ (min & max)

1 - 60 days (min & max)

COSC 4351
Estimating Exercise

Name:

Full Stack Web Applications Java, C#, Ruby, Python, or PhpS.

For the DVD_Collection Application below create a website in Java, C#, Ruby, Python, or PhpS for front end and a database backend as MS SQLSERVER, ORACLE, or MySQL database management systems (DBMS).

Estimate the LOC (lines of code that you will write): LOC
Estimate duration (in days): days
Estimate cost (in dollars): dollars

Metrics for the Size of a Product

Since LOC is unreliable, other Metrics can be considered

1 FFP (Files, Flows, Processes)

2 Function Points (based on # Inputs, Outputs, Inquiries, Master files, Interfaces)

3 COCOMO (algorithmic Cost Estimation Model – CASE tools)

Lines Of Code

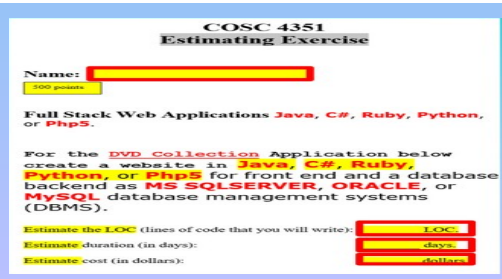
Estimation based on **LOC** is therefore doubly dangerous

- To start the **Estimation process**, ***LOC*** *in the* ***finished Product*** *must be* ***Estimated***

- The **LOC Estimate** is then used to ***Estimate*** *the* ***Cost*** *of the* ***product*** — an **uncertain input to an uncertain Cost Estimato**

DVDCollection

DVDCollection Web App – 50 – 1,000,000 LOC		(min & max)
	10 – 2,000,000 $	(min & max)
	1 - 60 days	(min & max)


COSC 4351
Estimating Exercise

Name:

500 points

Full Stack Web Applications Java, C#, Ruby, Python, or PhpS.

For the DVD Collection Application below create a website in Java, C#, Ruby, Python, or PhpS for front end and a database backend as MS SQLSERVER, ORACLE, or MySQL database management systems (DBMS).

Estimate the LOC (lines of code that you will write): LOC

Estimate duration (in days): days

Estimate cost (in dollars): dollars

Metrics for the Size of a Product

Metrics based on measurable quantities that can be **determined early** in the software life cycle

1. FFP (Files, Flows, Processes)

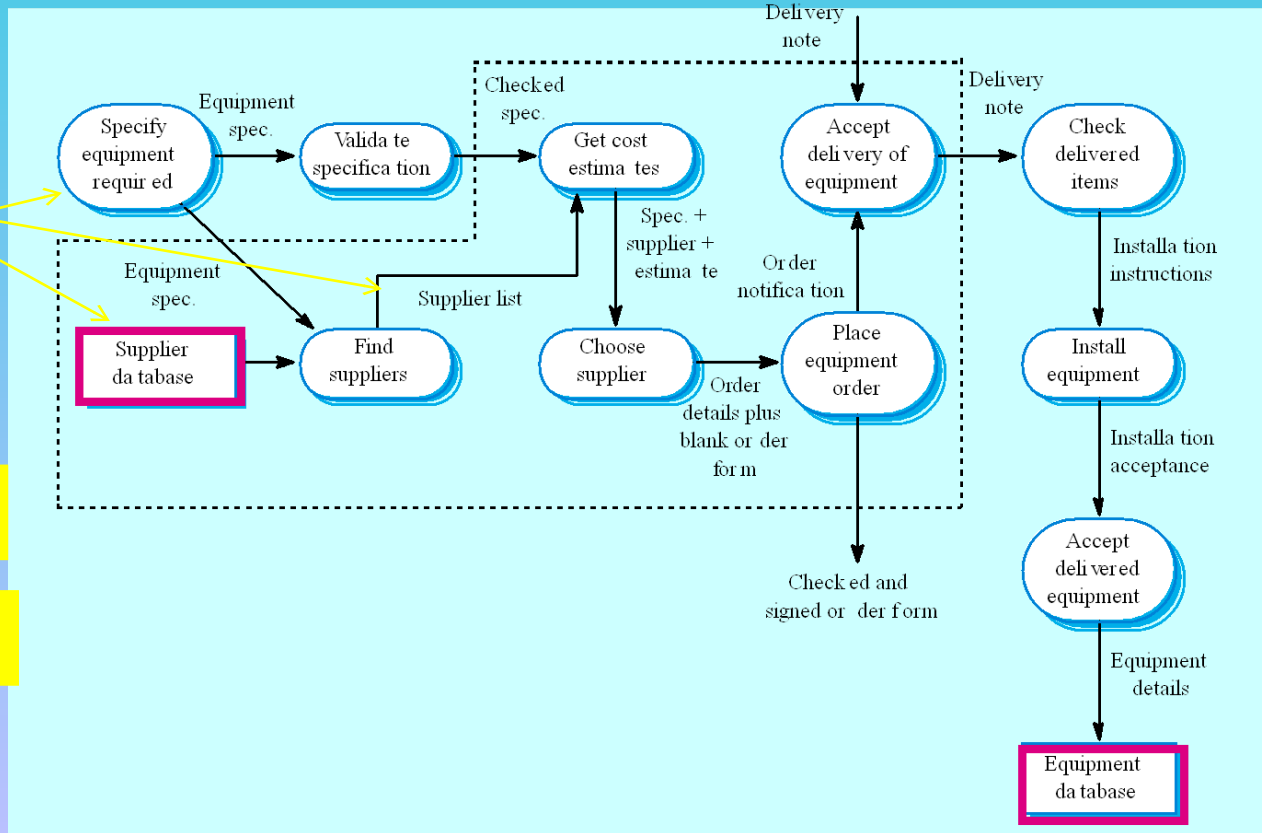
2. Function points (based on # **I**nputs, **O**utputs, **I**nquiries, **M**aster files, **I**nterfaces)

1. FFP Metric (Files, Flows, Processes)

Cost Estimation of medium-scale *data processing* Products

The three basic structural elements of data processing Products

- Files
- Flows
- Processes



How many F, F, P?

2, 13, 10

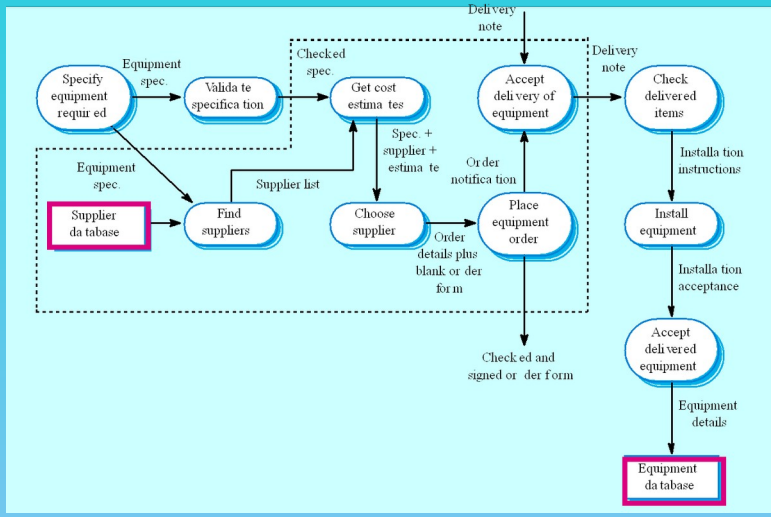
FFP Metric (Files, Flows, Processes)

Given the # of Files (*Fi*), Flows (*Fl*), and Processes (*Pr*)

- The **Size** (*S*), **Cost** (*C*) are given by

$$S = Fi + Fl + Pr$$

$$C = b \times S$$



The constant *b* (efficiency or productivity) **varies** from Organization to Organization

FFP Metric (Files, Flows, Processes)

The validity and reliability of the **FFP** Metric were demonstrated using a **purposive sample**

- However, the **FFP Metric** was never extended to include Databases

2. Function Points Metric

Based on the # of Inputs (*Inp*), Outputs (*Out*), Inquiries (*Inq*), Master Files (*Maf*), Interfaces (*Inf*)

For **any Product**, the Size in "Function Points" is given by

$$FP = 4 \times Inp + 5 \times Out + 4 \times Inq + 10 \times Maf + 7 \times Inf$$

This is an oversimplification of a 3-step process

Function Points

Step 1. Classify **each component of the Product** (*Inp*, *Out*, *Inq*, *Maf*, *Inf*) as Simple, Average, or Complex

- Assign the appropriate **#** of **F**unction **P**oints
- The sum gives **UFP** (**U**nadjusted **F**unction **P**oints)

	Level of Complexity		
Component	**Simple**	**Average**	**Complex**
Input item	3	4	6
Output item	4	5	7
Inquiry	3	4	6
Master file	7	10	15
Interface	5	7	10

Figure 9.3

Function Points

Step 2. Compute the Technical Complexity Factor (*TCF*)

- Assign a value from 0 ("not present") to 5 ("strong influence throughout") to each of 14 factors such as Transaction Rates, Portability

14 factors

1. Data communication
2. Distributed data processing
3. Performance criteria
4. Heavily utilized hardware
5. High transaction rates
6. Online data entry
7. End-user efficiency
8. Online updating
9. Complex computations
10. Reusability
11. Ease of installation
12. Ease of operation
13. Portability
14. Maintainability

Function Points

Add the 14 numbers

- This gives the total **D**egree of **I**nfluence (***DI***)

$$TCF = 0.65 + 0.01 \times DI$$

14 factors

1. Data communication
2. Distributed data processing
3. Performance criteria
4. Heavily utilized hardware
5. High transaction rates
6. Online data entry
7. End-user efficiency
8. Online updating
9. Complex computations
10. Reusability
11. Ease of installation
12. Ease of operation
13. Portability
14. Maintainability

The **T**echnical **C**omplexity **F**actor (***TCF***)

lies between **0.65** and **1.35**

Function Points

Step 3. The number of **F**unction **P**oints (*FP*) is then given by

$$FP = UFP \times TCF$$

Unadjusted **F**unction **P**oints × **T**echnical **C**omplexity **F**actor

Analysis of Function Points Metric

Function Points are usually **better** than **KDSI** — but there are some problems

- "Errors in excess of 800% counting **KDSI**, but *only* 200% in counting Function Points" [Jones, 1987]

Analysis of Function Points Metric

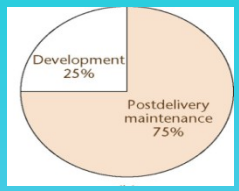
- We obtain **nonsensical** results from **Metrics**
  - **KDSI** per person month ?!?!?! and
  - **Cost** per source statement ?!?!?

	Assembler Version	Ada Version
Source code size	70 KDSI	25 KDSI
Development costs	$1,043,000	$590,000
KDSI per person-month	0.335	0.211
Cost per source statement	$14.90	$23.60
Function points per person-month	1.65	2.92
Cost per function point	$3,023	$1,170

- Cost per **Function Point** is **meaningful**

Maintenance Estimates: FFP, Function Points Metrics

- ***Postdelivery Maintenance*** can be **inaccurately** measured



- It is possible to make major changes without changing
  - The number of Files, Flows, and Processes (FFP);
  - The number of Inputs, Outputs, Inquiries, Master files, and Interfaces (Function Points)
- In theory, it is possible to change every Line Of Code without changing the # of Lines Of Code

Mark II Function Points Metric

This Metric was put forward to compute *UFP* **more accurately**
Unadjusted Function Points

We decompose **software** into component transactions, each consisting of **input**, **process**, and **output**

Mark II Function Points are widely used **all over the world**

Techniques of Cost Estimation

- Expert judgment by analogy
- Bottom-up approach
- Algorithmic Cost Estimation Models

Expert Judgment by Analogy

Experts compare the target **product** to completed **product**

need to track

- Guesses can lead to ***hopelessly incorrect*** **Cost Estimates**
- **Experts** may recollect completed **products** ***inaccurately***
- **Human Experts** have ***biases***

- However, the results of **Estimation** by a broad group of **Experts** **may be accurate**

Expert Judgment by Analogy

The Delphi technique is sometimes needed to achieve consensus

Bottom-up Approach

Break the **product** into smaller components

- The smaller components may be no easier to **Estimate**
- However, there are **process**-level **Costs**

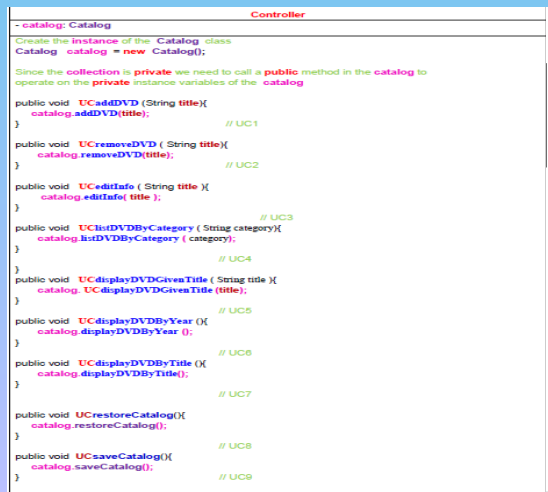
When using the **Object Oriented Paradigm**

- The independence of the **Classes** assists here
- However, the interactions among the **Classes** (coupling) **complicate** the **Estimation process**

thus, MVC

DVDCollection Application - ? Classes ? methods

????



```
Controller
- catalog: Catalog

Create the instance of the Catalog class
Catalog catalog = new Catalog();

Since the collection is private we need to call a public method in the catalog to
operate on the private instance variables of the catalog

public void UCaddDVD (String title){
    catalog.addDVD(title);
}                                   // UC1

public void UCremoveDVD ( String title){
    catalog.removeDVD(title);
}                                   // UC2

public void UCeditInfo ( String title ){
    catalog.editInfo( title );
}                                   // UC3

public void UClistDVDByCategory ( String category){
    catalog.listDVDByCategory ( category);
}                                   // UC4
}
public void UCdisplayDVDGivenTitle ( String title ){
    catalog. UCdisplayDVDGivenTitle (title);
}                                   // UC5

public void UCdisplayDVDByYear (){
    catalog.displayDVDByYear ();
}                                   // UC6

public void UCdisplayDVDByTitle (){
    catalog.displayDVDByTitle();
}                                   // UC7

public void UCrestoreCatalog(){
    catalog.restoreCatalog();
}                                   // UC8

public void UCsaveCatalog(){
    catalog.saveCatalog();
}                                   // UC9
```

2. Algorithmic Cost Estimation Models

A **Metric** (Function Points, FFP) is used as an input to a **Model** to compute **Cost** and **Duration**

- An Algorithmic Model is unbiased, and therefore superior to **Expert** opinion
- However, **Estimates** are **only as good as the underlying assumptions**

Examples

- SLIM **Model**
- Price S **Model**
- **CO**nstructive **CO**st **MO**del (**COCOMO**)

Intermediate COCOMO

COCOMO consists of three Models

- A macro-estimation Model for the product as a whole
- Intermediate COCOMO Model
- A micro-estimation Model that treats the product in detail

We examine Intermediate COCOMO Model(Boehm 1981)

Intermediate COCOMO

- **Step 1.** Estimate the length of the **product** in KDSI

WHAT?!?!

Impediment to Accurate Estimates: Human Factors

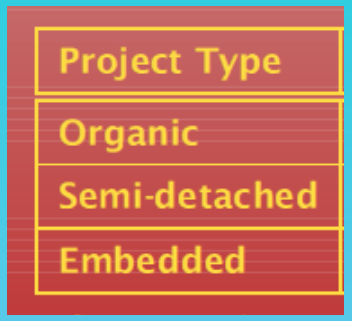
Sackman (**1968**) measured differences of up to 28 to 1 between pairs of programmers (*10 years* to beginners)

- He compared matched pairs of programmers with respect to
  - **Product** size 6 to 1

Intermediate COCOMO

Step 2. Estimate the **product Development** mode

- **Organic**,
- **S**emi-detached,
- **E**mbedded



Nominal Effort = a × (KDSI)^b

Example:

- Straightforward **product** ("**Organic**")

Intermediate COCOMO

Project Type	*a*	*b*
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

Step 3. Compute the **Nominal Effort**

Nominal Effort = $a \times (KDSI)^b$

Example:

- **Organic product**
- 12,000 delivered source statements (12 KDSI) (**Estimated**)

Nominal Effort = $3.2 \times (12)^{1.05}$ = **43 person-months**

Note:

If 43 person-months of **effort** are needed to **produce 12,000** Delivered Source Instructions, then on average each **Programmer** is turning out fewer than 300 Lines **O**f **C**ode a month (12,000/43).

Maintainable 12,000 Line product has to go through all the workflows of the life cycle. Thus, the **effort** is shared among many activities, including **coding**.

Intermediate COCOMO

Step 4.
Multiply the Nominal Effort by 15 software Development cost driver multipliers

Cost Drivers	Rating					
	Very Low	Low	Nominal	High	Very High	Extra High
Product attributes						
Required software reliablity	0.75	0.88	1.00	1.15	1.40	
Database size		0.94	1.00	1.08	1.16	
Product complexity	0.70	0.85	1.00	1.15	1.30	1.65
Computer attributes						
Execution time constraint			1.00	1.11	1.30	1.66
Main storage constraint			1.00	1.06	1.21	1.56
Virtual machine volatility*		0.87	1.00	1.15	1.30	
Computer turnaround time		0.87	1.00	1.07	1.15	
Personnel attributes						
Analyst capabilities	1.46	1.19	1.00	0.86	0.71	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Programmer capability	1.42	1.17	1.00	0.86	0.70	
Virtual machine experience*	1.21	1.10	1.00	0.90		
Programming language experience	1.14	1.07	1.00	0.95		
Project attributes						
Use of modern programming practices	1.24	1.10	1.00	0.91	0.82	
Use of software tools	1.24	1.10	1.00	0.91	0.83	
Required development schedule	1.23	1.08	1.00	1.04	1.10	

*For a given software product, the underlying virtual machine is the complex of hardware and software (operating system, database management system) it calls on to accomplish its task.

Intermediate COCOMO – Example

Microprocessor-based **communications processing software** for electronic funds transfer network with ***high*** Reliability, Performance, Development schedule, and Interface requirements

Step 1. Estimate the length of the product

- 10,000 delivered source instructions (10 KDSI)

Step 2. Estimate the product development mode

- Complex ("Embedded") mode

Project Type	*a*	*b*
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

Intermediate COCOMO – Example

Step 3. Compute the Nominal Effort

- Nominal Effort = $2.8 \times (10)^{1.20}$ = **44 person-months**

Step 4. Multiply the Nominal Effort by 15 software development cost multipliers

- Product of effort multipliers = 1.35 (**see table on next slide**)
- **Estimated effort** for project is therefore 1.35 × **44** = **59** person-months

Embedded	2.8	1.20

Intermediate COCOMO – Example

- Software development effort multipliers

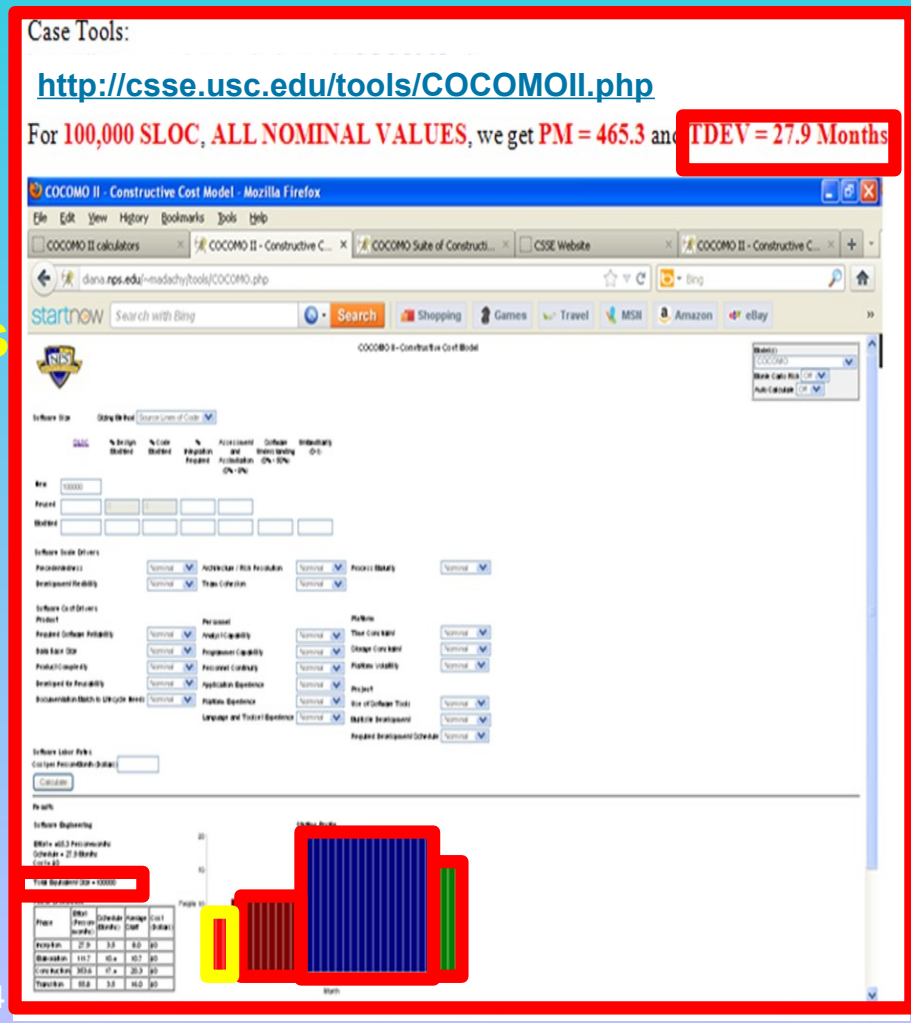
Cost Drivers	Situation	Rating	Effort Multiplier
Required software reliability	Serious financial consequences of software fault	High	1.15
Database size	20,000 bytes	Low	0.94
Product complexity	Communications processing	Very high	1.30
Execution time constraint	Will use 70% of available time	High	1.11
Main storage constraint	45K of 64K store (70%)	High	1.06
Virtrual machine volatility	Based on commercial microprocessor hardware	Nominal	1.00
Computer turnaround time	Two hour average turnaround time	Nominal	1.00
Analyst capabilities	Good senior analysts	High	0.86
Applications experience	Three years	Nominal	1.00
Programmer capability	Good senior programmers	High	0.86
Virtual machines experience	Six months	Low	1.10
Programming language experience	Twelve months	Nominal	1.00
Use of modern programming practices	Most techniques in use over one year	High	0.91
Use of software tools	At basic minicomputer tool level	Low	1.10
Required development schedule	Nine months	Nominal	1.00

Microprocessor-based communications processing SOFTWARE for electronic funds transfer network with *high* Reliability, Performance, Development schedule, and Interface requirements

Intermediate COCOMO

Estimated effort for project (**59 person-months**) is used as input for additional formulas for:

- Dollar **costs**
- Development **schedules**
- Phase and activity **distributions**
- Computer **costs**
- Annual maintenance **costs**
- Related **items**



Intermediate COCOMO

- Intermediate COCOMO has been validated with respect to a **broad sample**
- Actual values are within 20% of **predicted values** about 68% of the time
  - Intermediate **COCOMO** was the most accurate **Estimation Method** of its time
- Major problem
  - **If** the **Estimate** of the *number of **L**ines **O**f **C**odes* of the **target product** is incorrect, then **everything** is incorrect

COCOMO II

- **1995** extension to 1981 **COCOMO** (Boehm **2000**) that incorporates
  - Object orientation - **OOP**
  - Modern life-cycle models - **Agile**
  - Rapid prototyping
  - Fourth Generation Languages – **4GL**
  - COTS software - **Reuse**
- **COCOMO II** is far more complex than the first version

COCOMO II

- There are three different **Models**
  - **Application composition Model** for the **early phases**
    - Based on **F**eature **P**oints (similar to **F**unction **P**oints)
  - **Early Design Model**
    - Based on **F**unction **P**oints
  - **Post-Architecture Model**
    - Based on **F**unction **P**oints or **KDSI**

COCOMO II

- The underlying **COCOMO** effort **Model** is

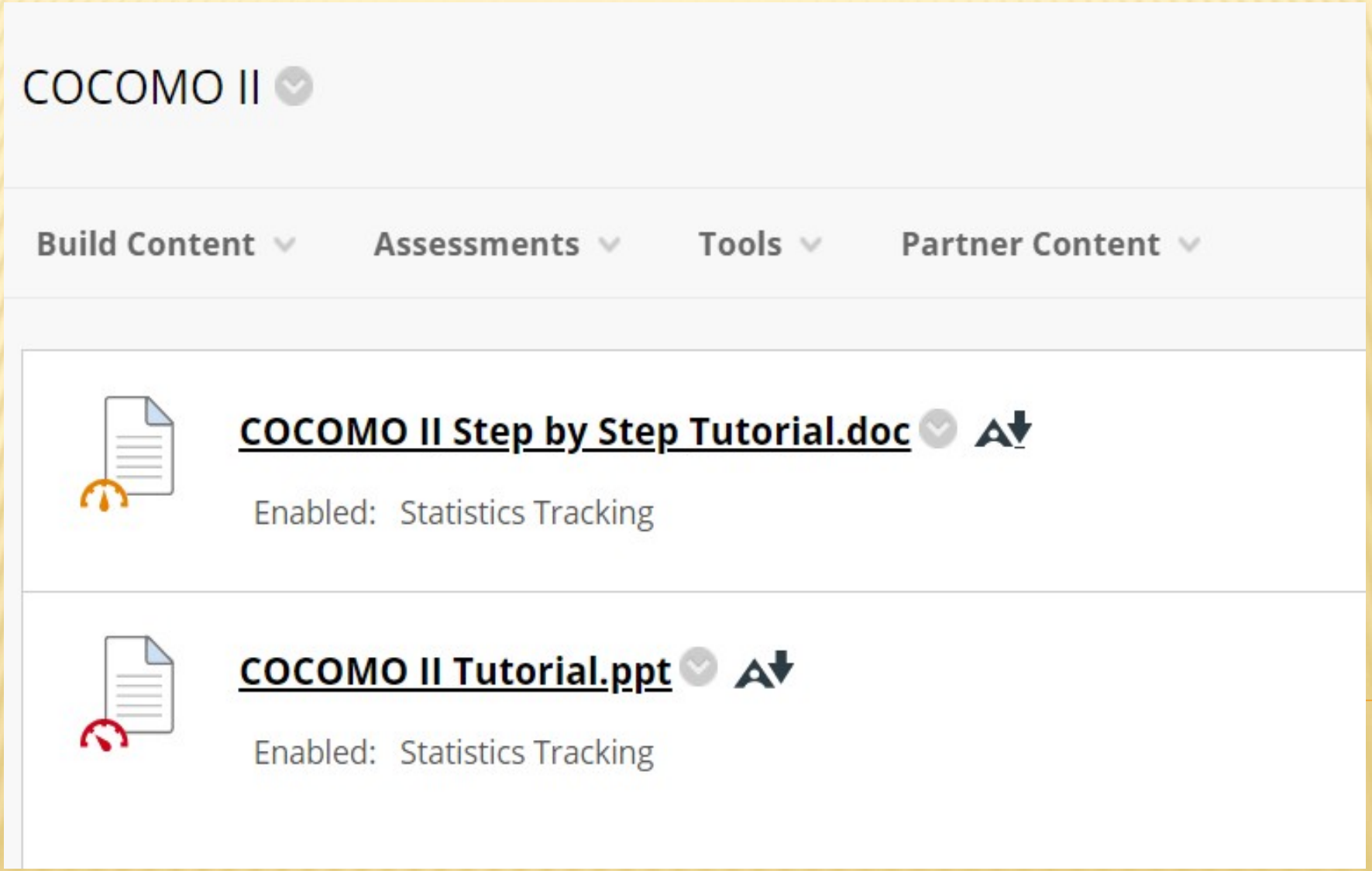
$$\text{Nominal Effort} = a \times (\text{size})^b$$

Project Type	*a*	*b*
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

  - **Intermediate** COCOMO
    - Three values for (*a*, *b*)
  - COCOMO II
    - *b* varies depending on the values of certain parameters

- COCOMO II supports **Reuse**

COCOMO II TUTORIAL



COCOMO II ESTIMATES FOR DELIVERY SYSTEM

Use COCOMO II model to estimate the effort and schedule for **medium scale** data storage and retrieval software application for an overnight delivery trucking company **NOT High Risk**.

- Approximately **934 Unadjusted Function Points UFP**. (Hint need to convert to **SLOC**).
  (See Table 1 60 SLOC/FP)
- Prototyping and a basic architectural design have been complete, so use the **Post-Architectural model**.
- Certain scale and effort multipliers can be obtained by the following scenario (others may be assumed **nominal**).

The software company has recently had to make a few cutbacks to personnel and a few highly experienced personnel have jumped ship for greener pastures. The remaining employees, all **located in the same building** (SITE – **Very High**) are all **college buddies** (TEAM – **Very High**) and almost **communicate by telepathy** (PCON – **Extra High**). Although the group is extremely **fluent in Java** (AEXP – **Very High**) and **database architecture** (PREC– **Very High**), the client insists the program be **written in C++** (LTEX - **Low**). The team has been asked to complete the project on a **slightly accelerated schedule** (TIME – **High**).

COCOMO II ESTIMATES FOR DELIVERY SYSTEM BY HAND ESTIMATES

Approximately **934 Unadjusted Function Points**. (Hint need to convert to **SLOC**).

934 UFPs * 60 C++ SLOC/UFP = 56,040 SLOC (Source Lines Of Code)

Using Post-Architectural mo

- Approximately 934 Unadjusted Function Points UFP. (Hint need to convert to SLOC). (See Table 1 60 SLOC/FP)

$$PM_{NS} = A \times (Size)^{E} \times \prod_{i=1}^{n} EM_i$$

$$\text{where } E = B + 0.01 \times \sum_{j=1}^{5} SF_j$$

College buddies and almost communicate by te

TEAM (**SF**) = **Very High(1)**

PCON (**EM**) = **Extra High (0.81)**

Table 1. COCOMO II Cost Drivers

	Abr.	Range							Name
Scale Factor	PREC	5	4	3	2	1	0		Precendentedness
	FLEX	5	4	3	2	1	0		Development Flexibility
	RESL	5	4	3	2	1	0		Architecture and Risk Resolution
	TEAM	5	4	3	2	1	0		Team cohesion
	PMAT	5	4	3	2	1	0		Process Maturity
		Extra Low	Very Low	Low	Nominal	High	Very High	Extra High	
Effort Multipliers (Cost Drivers)	RELY	0.82	0.82	0.92	1.00	1.10	1.26	1.26	Required Software
	DATA	0.90	0.90	0.90	1.00	1.14	1.28	1.28	Data Base Size
	CPLX	0.73	0.73	0.87	1.00	1.17	1.34	1.74	Product Complexity
	RUSE	0.95	0.95	0.95	1.00	1.07	1.15	1.24	Required Reusability
	DOCU	0.81	0.81	0.91	1.00	1.11	1.23	1.23	Documentation Match to Life-cycle Needs
	TIME	1.00	1.00	1.00	1.00	1.11	1.29	1.63	Time Constraint
	STOR	1.00	1.00	1.00	1.00	1.05	1.17	1.46	Storage Constraint
	PVOL	0.87	0.87	0.87	1.00	1.15	1.30	1.30	Platform Volatility
	ACAP	1.42	1.42	1.19	1.00	0.85	0.71	0.71	Analyst Capability
	PCAP	1.34	1.34	1.15	1.00	0.88	0.76	0.76	Programmer Capability
	AEXP	1.22	1.22	1.10	1.00	0.88	0.81	0.81	Applications Experience
	PEXP	1.19	1.19	1.09	1.00	0.91	0.85	0.85	Platform Experience
	LTEX	1.20	1.20	1.09	1.00	0.91	0.84	0.84	Language and Tool Experience
	PCON	1.29	1.29	1.12	1.00	0.90	0.81	0.81	Personnel Continuity
	TOOL	1.17	1.17	1.09	1.00	0.90	0.78	0.78	Use of Software Tools
	SITE	1.22	1.22	1.09	1.00	0.93	0.86	0.86	Multi-Site Development
	SCED	1.43	1.43	1.12	1.00	1.00	1.00	1.00	Required Development Schedule
									Product / Platform / Personnel / Project

The software company has recently had to make a few cutbacks to personnel and a few highly experienced personnel have jumped ship for greener pastures. The remaining employees, all located in the same building (SITE – **Very High**) are all college buddies (TEAM – **Very High**) and almost communicate by telepathy (PCON – **Extra High**). Although the group is extremely fluent in Java (AEXP – **Very High**) and database architecture (PREC– **Very High**), the client insists the program be written in **C++** (LTEX - **Low**). The team has been asked to complete the project on a **slightly accelerated schedule** (TIME – **High**).

COCOMO II ESTIMATES FO
BY HAND EST

Written in **C++** -

LTEX (**EM**) = Low(1.09)

Extremely fluent database architecture,

PREC (**SF**) = **Very High** (**1**)

AEXP (**EM**) = Very High(0.81)

Slightly accelerated schedule

TIME (**EM**) = High(1.11)

Same Location

SITE (**EM**) = Very High(0.86)

A = 2.94 (calibrated value)

TEAM (SF) = Very High(1)

Table 1. COCOMO II Cost Drivers

	Abr.	Range						Name
Scale Factor	PREC	5	4	3	2	1	0	Precendentedness
	FLEX	5	4	3	2	1	0	Development Flexibility
	RESL	5	4	3	2	1	0	Architecture and Risk Resolution
	TEAM	5	4	3	2	1	0	Team cohesion
	PMAT	5	4	3	2	1	0	Process Maturity
		Extra Low	Very Low	Low	Nominal	High	Very High	Extra High	
Effort Multipliers (Cost Drivers)	RELY	0.82	0.82	0.92	1.00	1.10	1.26	1.26	Required Software
	DATA	0.90	0.90	0.90	1.00	1.14	1.28	1.28	Data Base Size
	CPLX	0.73	0.73	0.87	1.00	1.17	1.34	1.74	Product Complexity
	RUSE	0.95	0.95	0.95	1.00	1.07	1.15	1.24	Required Reusability
	DOCU	0.81	0.81	0.91	1.00	1.11	1.23	1.23	Documentation Match to Life-cycle Needs
	TIME	1.00	1.00	1.00	1.00	1.11	1.29	1.63	Time Constraint
	STOR	1.00	1.00	1.00	1.00	1.05	1.17	1.46	Storage Constraint
	PVOL	0.87	0.87	0.87	1.00	1.15	1.30	1.30	Platform Volatility
	ACAP	1.42	1.42	1.19	1.00	0.85	0.71	0.71	Analyst Capability
	PCAP	1.34	1.34	1.15	1.00	0.88	0.76	0.76	Programmer Capability
	AEXP	1.22	1.22	1.10	1.00	0.88	0.81	0.81	Applications Experience
	PEXP	1.19	1.19	1.09	1.00	0.91	0.85	0.85	Platform Experience
	LTEX	1.20	1.20	1.09	1.00	0.91	0.84	0.84	Language and Tool Experience
	PCON	1.29	1.29	1.12	1.00	0.90	0.81	0.81	Personnel Continuity
	TOOL	1.17	1.17	1.09	1.00	0.90	0.78	0.78	Use of Software Tools
	SITE	1.22	1.22	1.09	1.00	0.93	0.86	0.86	Multi-Site Development
	SCED	1.43	1.43	1.12	1.00	1.00	1.00	1.00	Required Development Schedule

Product | Platform | Personnel | Project

E = **0.91** + 0.01 x **Σ SFj(j = 1 to 5)** = 0.91 + 0.01 x (**1**+3+3+**1**+3) = 1.02

EM = 0.81 x 1.09 x 0.81 x 1.11 x 0.86 = **0.6827**

PCON **LTEX** **AEXP** **TIME** **SITE**

$$PM_{NS} = A \times (Size)^{E} \times \prod_{i=1}^{n} EM_i$$

Therefore:

PM_{NS}= 2.94 x $56.04^{1.02}$ x 0.6827 = 121.9 person-months

$$\text{where } E = B + 0.01 \times \sum_{j=1}^{5} SF_j$$

COCOMO II ESTIMATES FOR DELIVERY SYSTEM BY HAND ESTIMATES

• Recall that the total effort for **PM(Person Months) is 121.9 person months** (based upon a nominal schedule)

• Now determine the **nominal development time**

TDEV= [**C** x (**PM**)^F]

where **C = 3.67** and **D = 0.28**

$$TDEV_{NS} = C \times (PM_{NS})^{F}$$

$$\text{where } F = D + 0.2 \times 0.01 \times \sum_{j=1}^{5} SF_j$$

F = 0.28 + 0.2 x 0.01 x (1+3+3+1+3) = 0.302

- **C = 3.67**
- **PM_{NS} = from nominal-schedule effort estimation**
- **D = 0.28**
- **SF: Scale Factors (5 for both models)**

Therefore:

T_{DEV}(calendar time in months) = 3.67 (121.9)^0.302 = 15.7 calendar months

COCOMO II ESTIMATES FOR DELIVERY SYSTEM BY HAND ESTIMATES

Staffing = PM / TDEV = 121.9 / 15.7 = 7.8 or **8 persons**

PM(Person Months)

TDEV(calendar time in months)

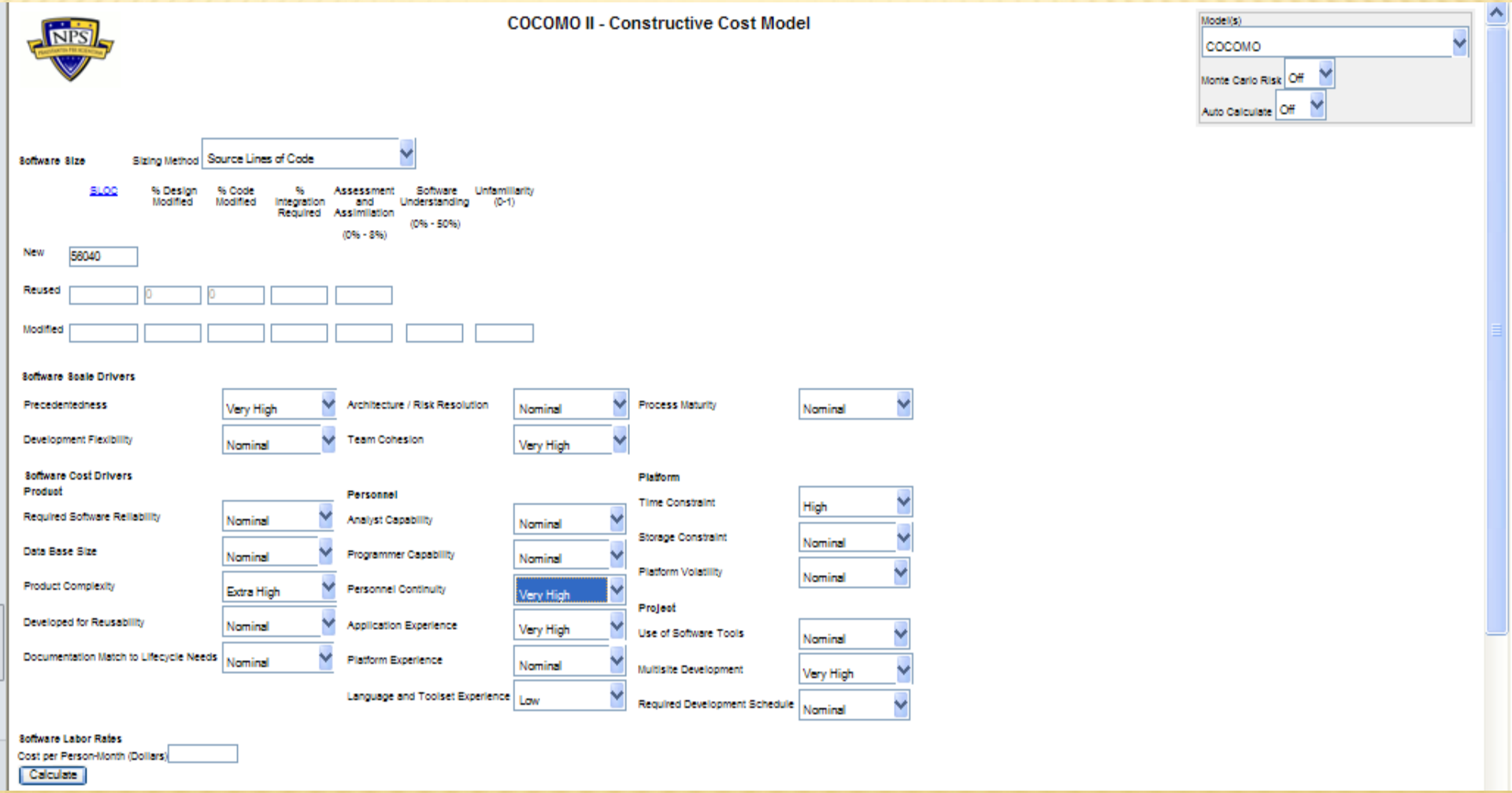
$$TDEV_{NS} = C \times (PM_{NS})^{F}$$

$$\text{where } F = D + 0.2 \times 0.01 \times \sum_{j=1}^{5} SF_j$$

- **C = 3.67**
- **PM_{NS} = from nominal-schedule effort estimation**
- **D = 0.28**
- **SF: Scale Factors (5 for both models)**

COCOMO II ESTIMATES FOR DELIVERY SYSTEM CASE TOOL ESTIMATES

http://csse.usc.edu/tools/COCOMOII.php



COCOMO II ESTIMATES FOR DELIVERY SYSTEM CASE TOOL ESTIMATES

Results

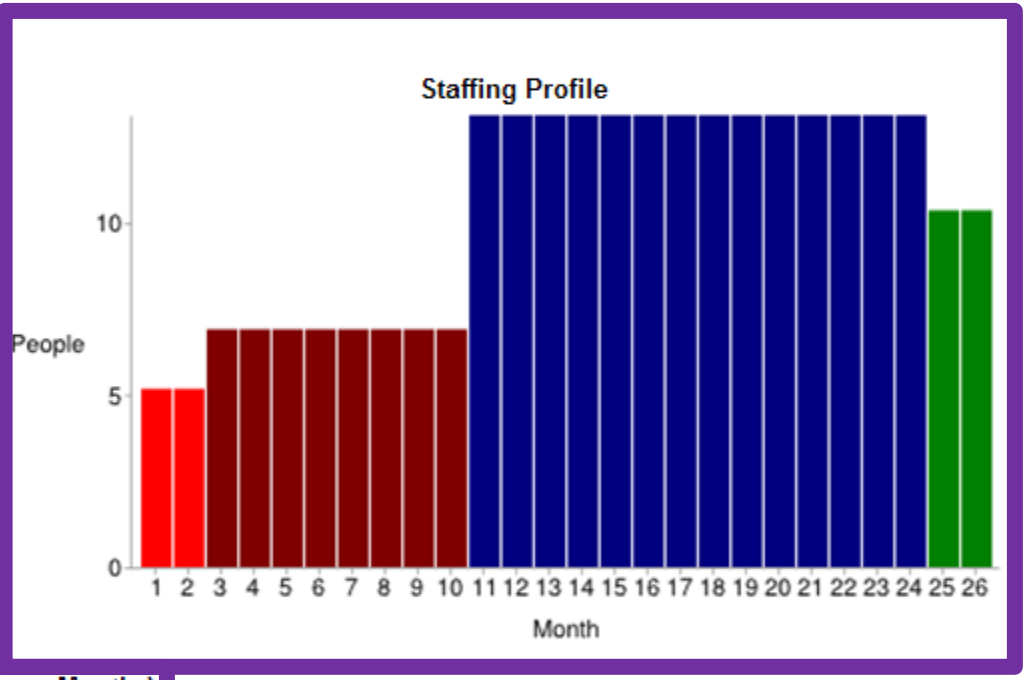
Software Engineering

Effort = 242.3 Person-months
Schedule = 22.5 Months
Cost = $0

Total Equivalent Size =56040

Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	14.5	2.8	5.2	$0
Elaboration	58.1	8.4	6.9	$0
Construction	184.1	14.0	13.1	$0
Transition	29.1	2.8	10.4	$0



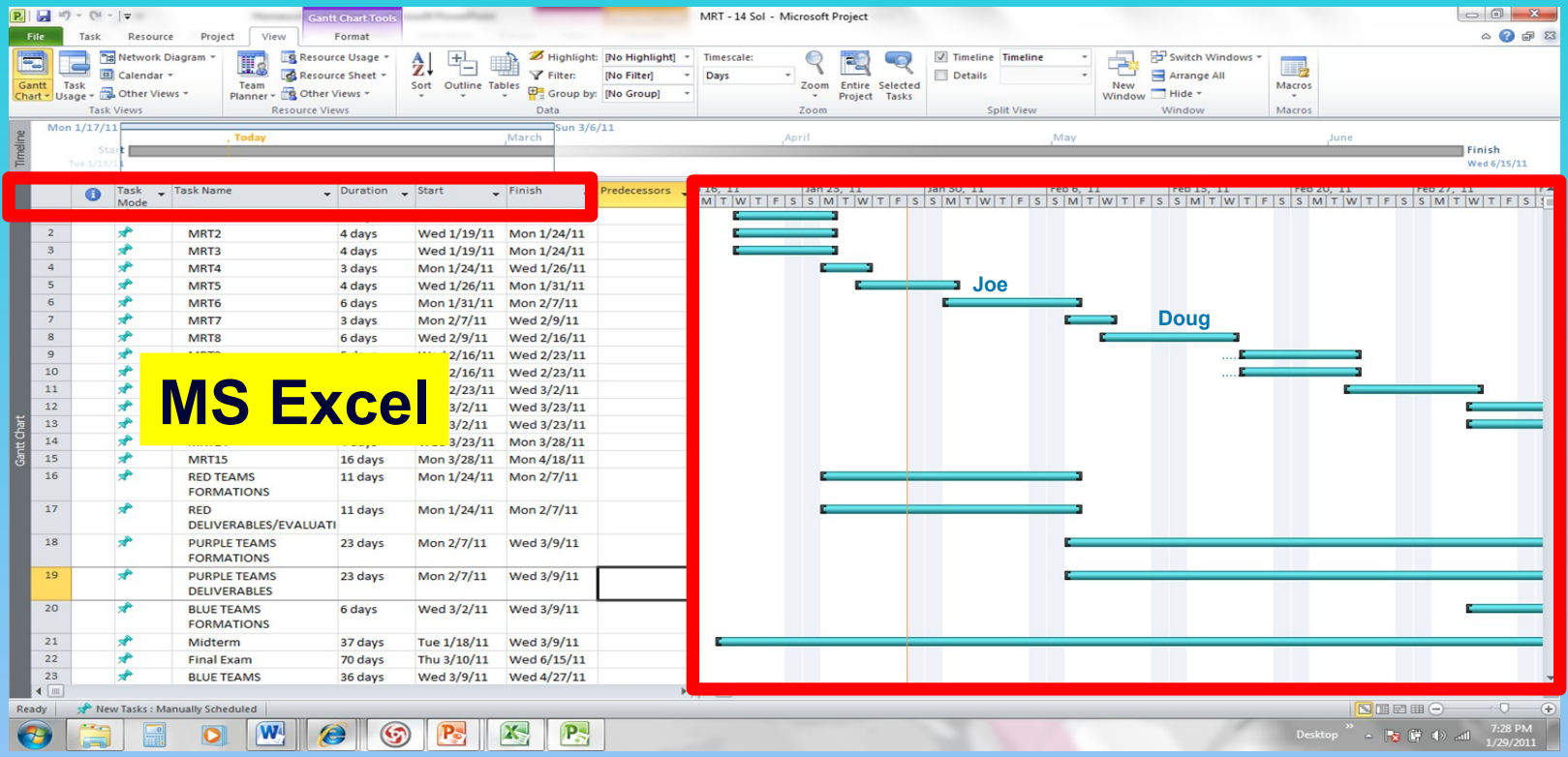
Software Effort Distribution for RUP/MBASE (Person-Months)

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	2.0	7.0	18.4	4.1
Environment/CM	1.5	4.7	9.2	1.5
Requirements	5.5	10.5	14.7	1.2
Design	2.8	20.9	29.5	1.2
Implementation	1.2	7.6	62.6	5.5
Assessment	1.2	5.8	44.2	7.0
Deployment	0.4	1.7	5.5	8.7

Your output file is http://diana.nps.edu/~madachy/tools/data/COCOMO_March_14_2012_17_06_31_648423.txt

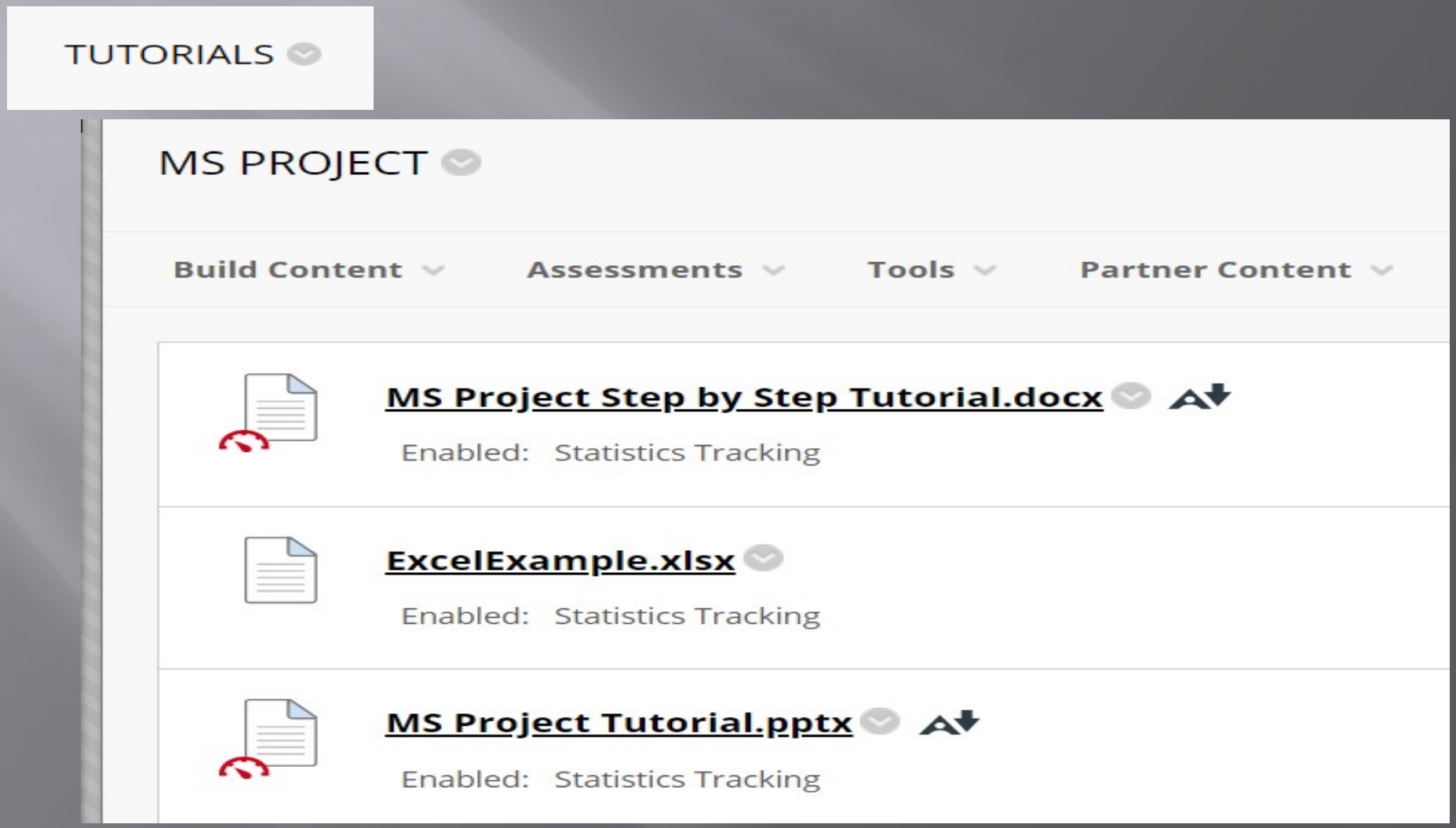
Tracking Duration and Cost Estimates

Whatever Estimation Method used, careful Tracking is vital



MS Project **Other CASE tools**

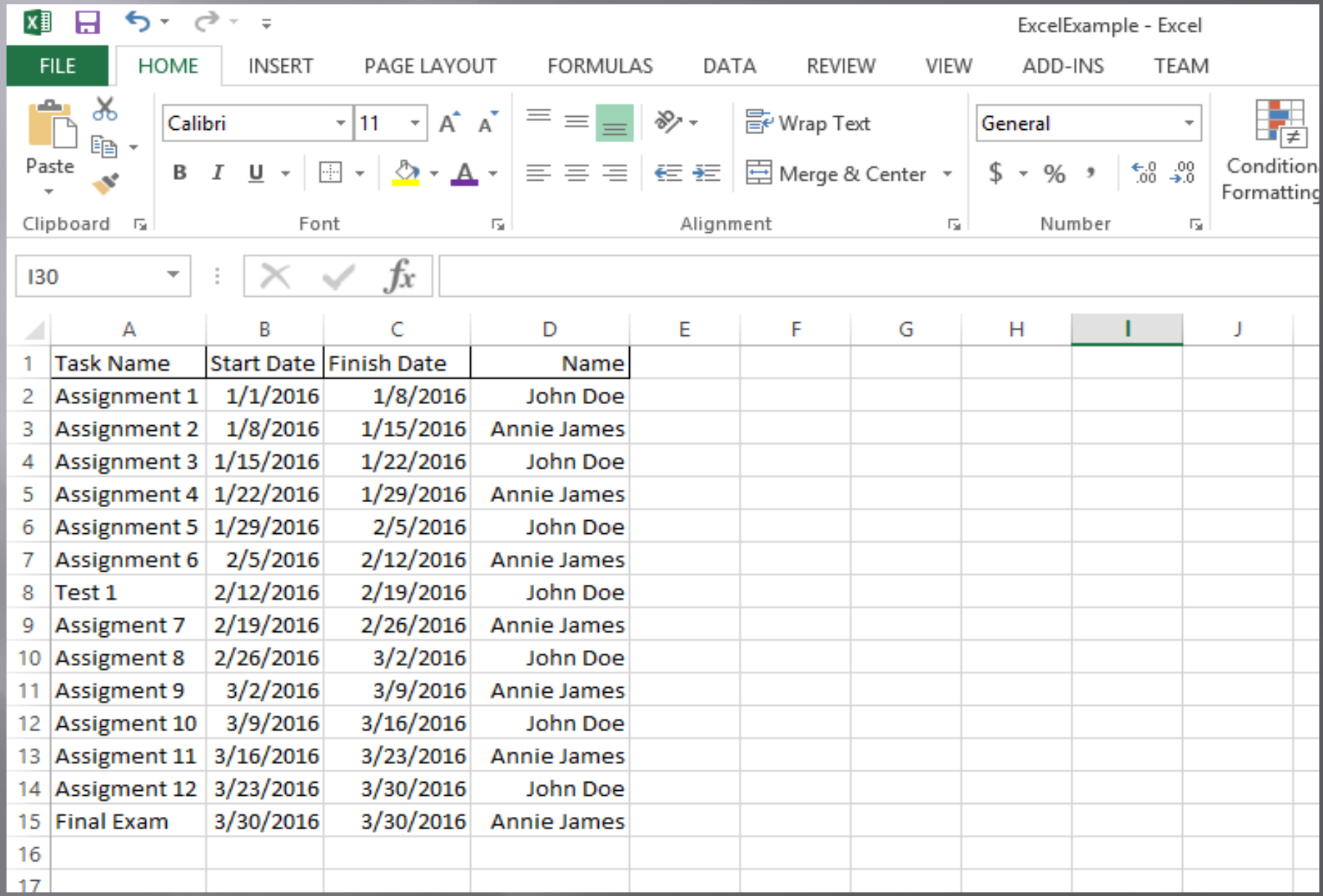
MS PROJECT TUTORIAL



Objectives

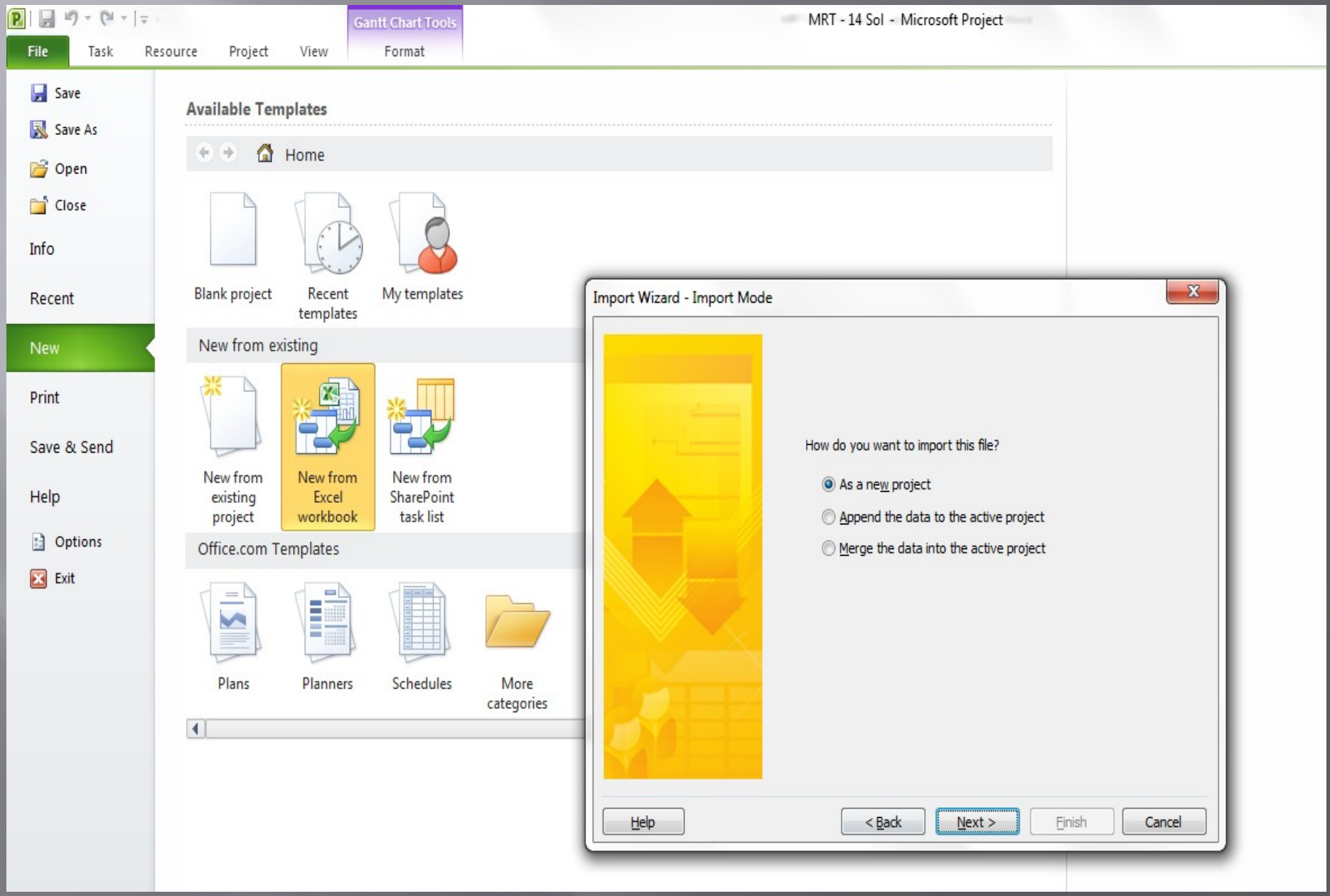
- Review Basic **Project Management** principles
- Review **MS Project**
- Create a project in MS Project using Excel data

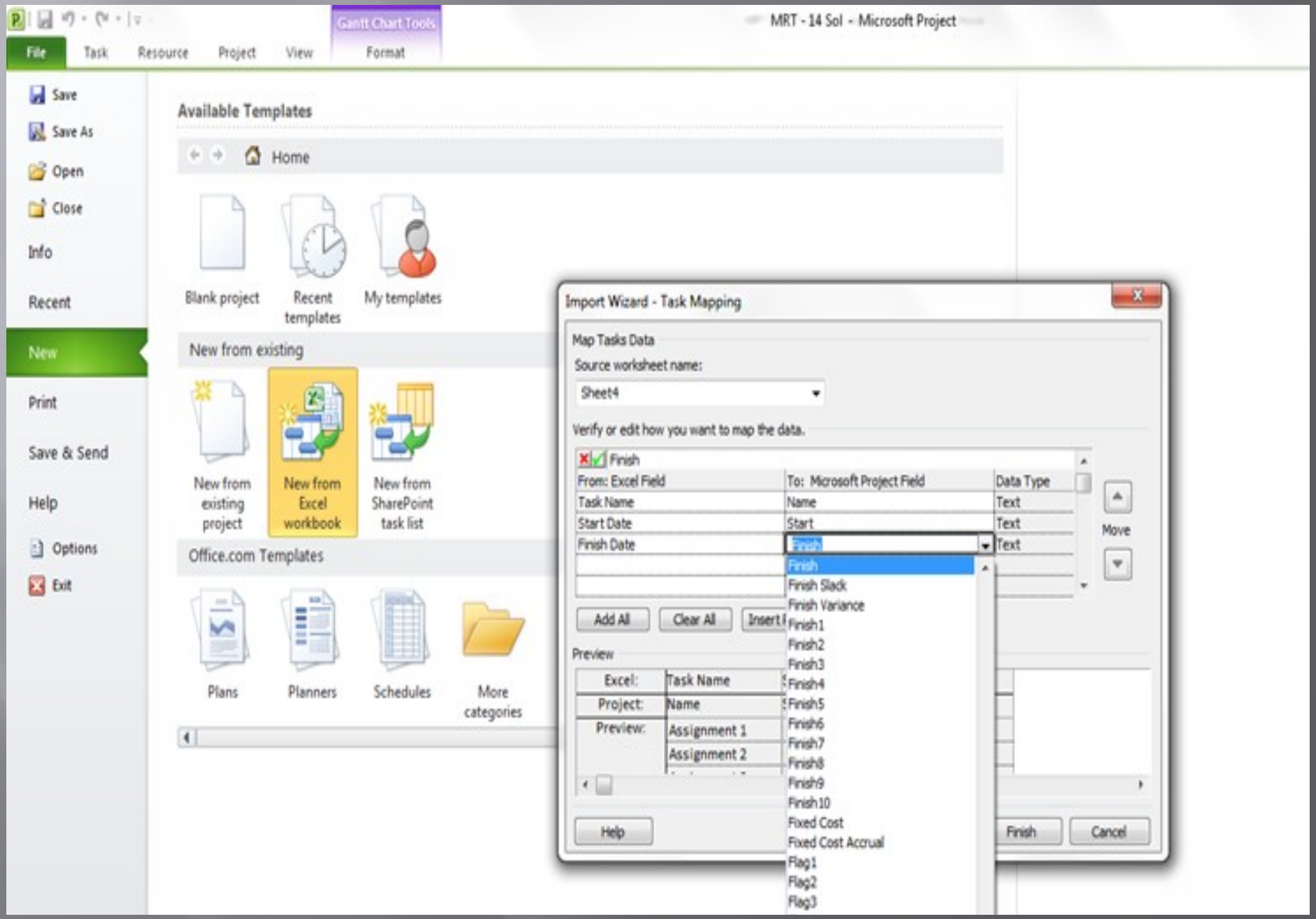
- Create the **Excel** spreadsheet

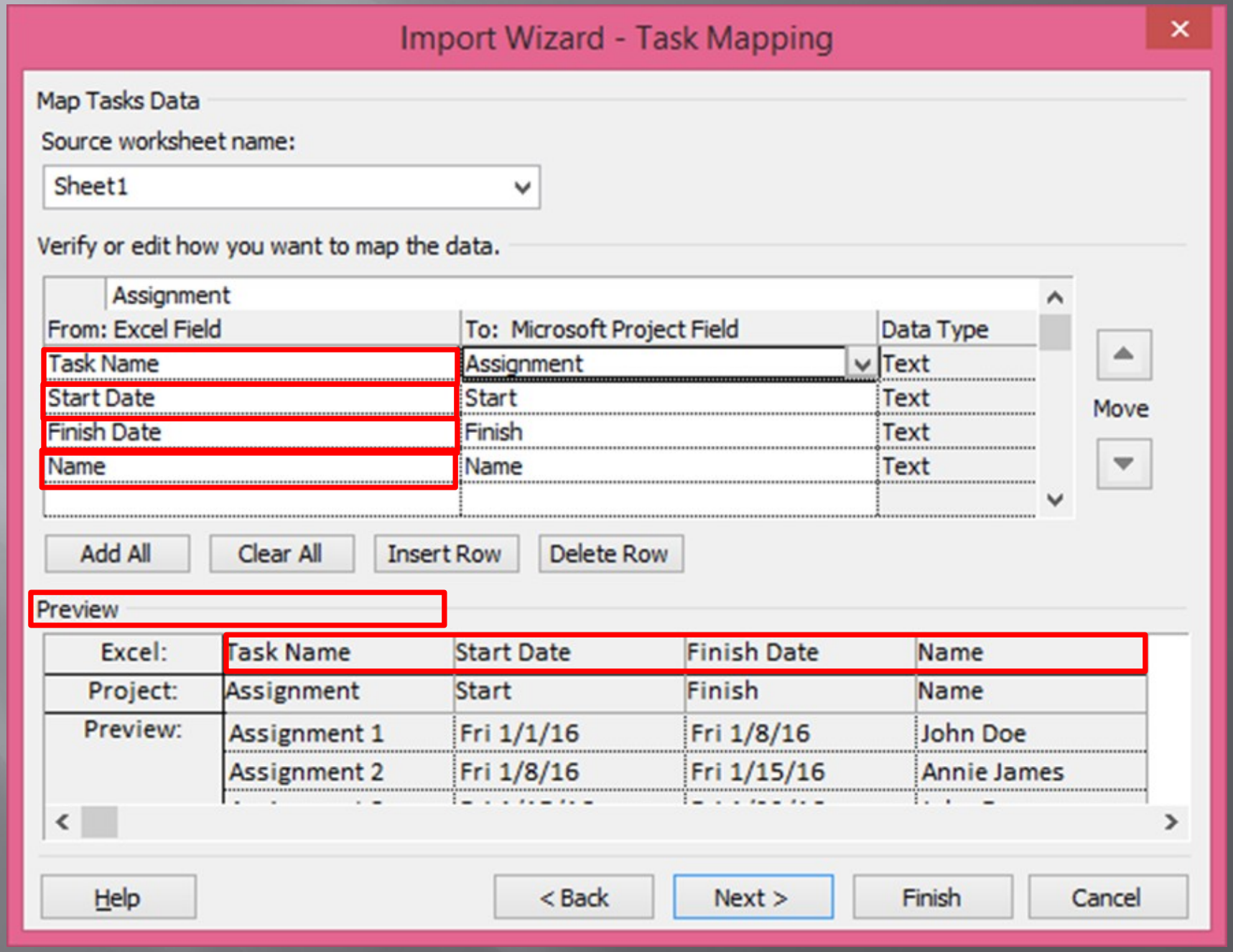


	A	B	C	D	E	F	G	H	I	J
1	Task Name	Start Date	Finish Date	Name						
2	Assignment 1	1/1/2016	1/8/2016	John Doe						
3	Assignment 2	1/8/2016	1/15/2016	Annie James						
4	Assignment 3	1/15/2016	1/22/2016	John Doe						
5	Assignment 4	1/22/2016	1/29/2016	Annie James						
6	Assignment 5	1/29/2016	2/5/2016	John Doe						
7	Assignment 6	2/5/2016	2/12/2016	Annie James						
8	Test 1	2/12/2016	2/19/2016	John Doe						
9	Assigment 7	2/19/2016	2/26/2016	Annie James						
10	Assigment 8	2/26/2016	3/2/2016	John Doe						
11	Assigment 9	3/2/2016	3/9/2016	Annie James						
12	Assigment 10	3/9/2016	3/16/2016	John Doe						
13	Assigment 11	3/16/2016	3/23/2016	Annie James						
14	Assigment 12	3/23/2016	3/30/2016	John Doe						
15	Final Exam	3/30/2016	3/30/2016	Annie James						
16										
17										

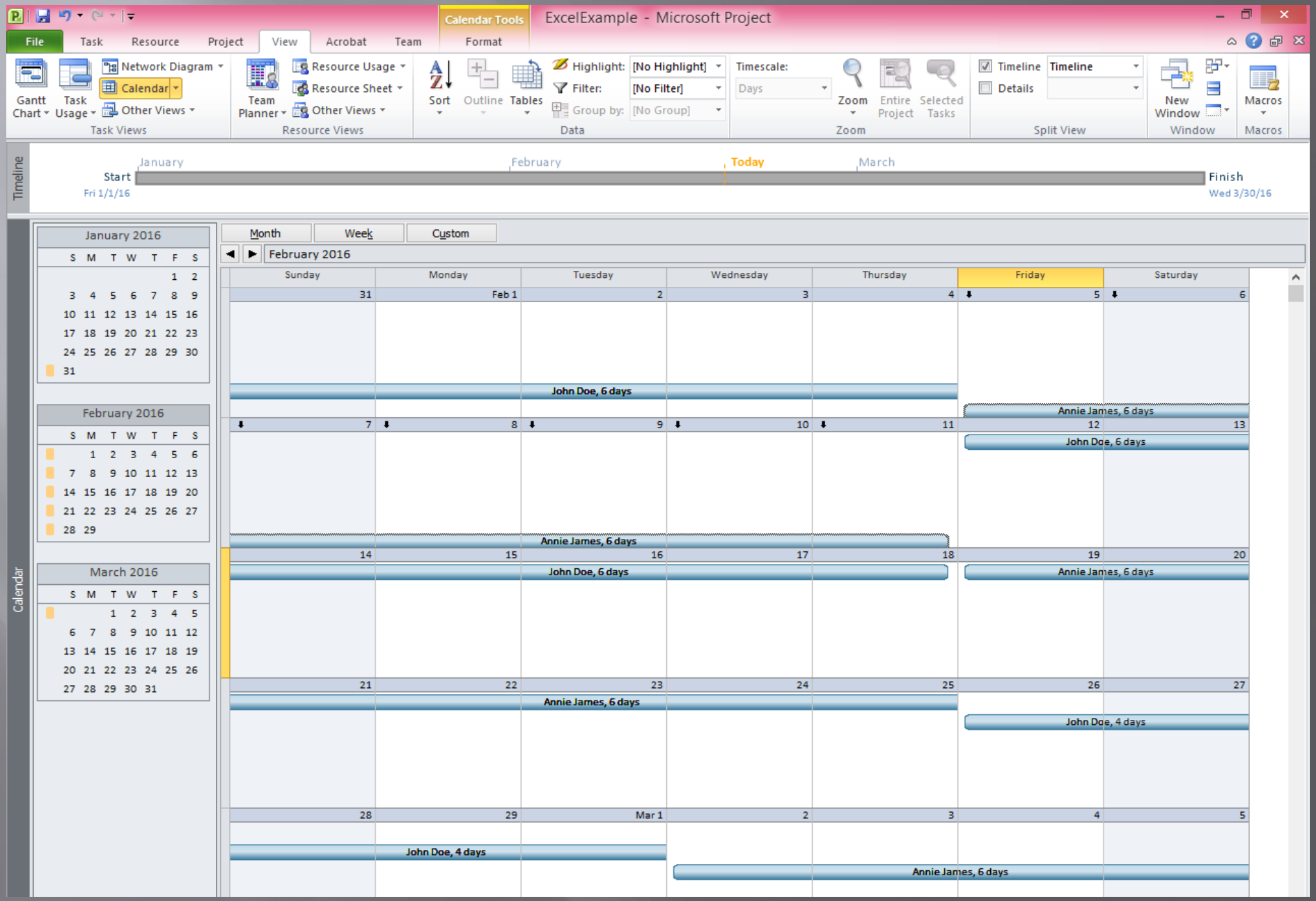
Import the **spreadsheet** into Project



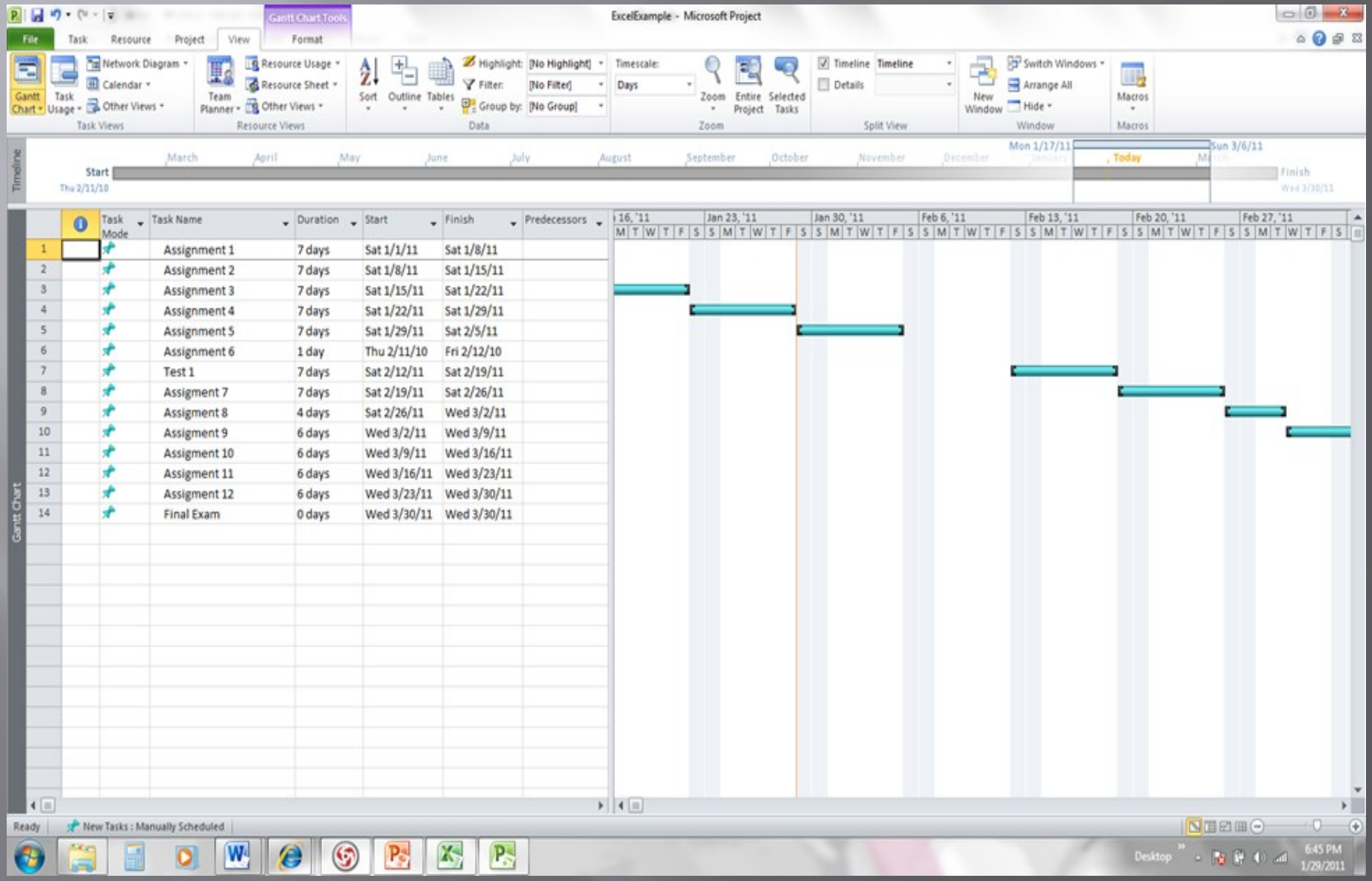

MRT - 14 Sol - Microsoft Project
Gantt Chart Tools
File
Task
Resource
Project
View
Format
Save
Save As
Open
Close
Info
Recent
New
Print
Save & Send
Help
Options
Exit
Available Templates
Home
Blank project
Recent templates
My templates
New from existing
New from existing project
New from Excel workbook
New from SharePoint task list
Office.com Templates
Plans
Planners
Schedules
More categories
Import Wizard - Task Mapping
Map Tasks Data
Source worksheet name:
Sheet4
Verify or edit how you want to map the data.
Finish
From: Excel Field
To: Microsoft Project Field
Data Type
Task Name
Name
Text
Start Date
Start
Text
Finish Date
Finish
Text
Move
Add All
Clear All
Preview
Excel:
Task Name
Project:
Name
Preview:
Assignment 1
Assignment 2
Finish
Finish Slack
Finish Variance
Finish1
Finish2
Finish3
Finish4
Finish5
Finish6
Finish7
Finish8
Finish9
Finish10
Fixed Cost
Fixed Cost Accrual
Flag1
Flag2
Flag3
Help
Finish
Cancel


Import Wizard - Task Mapping
Map Tasks Data
Source worksheet name:
Sheet1
Verify or edit how you want to map the data.
Assignment
From: Excel Field
To: Microsoft Project Field
Data Type
Task Name
Assignment
Text
Start Date
Start
Text
Finish Date
Finish
Text
Name
Name
Text
Move
Add All
Clear All
Insert Row
Delete Row
Preview
Excel:
Task Name
Start Date
Finish Date
Name
Project:
Assignment
Start
Finish
Name
Preview:
Assignment 1
Fri 1/1/16
Fri 1/8/16
John Doe
Assignment 2
Fri 1/8/16
Fri 1/15/16
Annie James
Help
< Back
Next >
Finish
Cancel

Calendar View



Gantt Chart


ExcelExample - Microsoft Project
Gantt Chart Tools
File
Task
Resource
Project
View
Format
Gantt Chart
Task Usage
Network Diagram
Calendar
Other Views
Team Planner
Resource Usage
Resource Sheet
Other Views
Sort
Outline
Tables
Highlight: [No Highlight]
Filter: [No Filter]
Group by: [No Group]
Timescale:
Days
Zoom
Entire Project
Selected Tasks
Timeline
Timeline
Details
New Window
Switch Windows
Arrange All
Hide
Macros
Task Views
Resource Views
Data
Zoom
Split View
Window
Macros
Timeline
Start
Thu 2/11/10
March
April
May
June
July
August
September
October
November
December
January
Today
Mon 1/17/11
Sun 3/6/11
Finish
Wed 3/30/11
Gantt Chart
Task Mode
Task Name
Duration
Start
Finish
Predecessors
1 Assignment 1 7 days Sat 1/1/11 Sat 1/8/11
2 Assignment 2 7 days Sat 1/8/11 Sat 1/15/11
3 Assignment 3 7 days Sat 1/15/11 Sat 1/22/11
4 Assignment 4 7 days Sat 1/22/11 Sat 1/29/11
5 Assignment 5 7 days Sat 1/29/11 Sat 2/5/11
6 Assignment 6 1 day Thu 2/11/10 Fri 2/12/10
7 Test 1 7 days Sat 2/12/11 Sat 2/19/11
8 Assigment 7 7 days Sat 2/19/11 Sat 2/26/11
9 Assigment 8 4 days Sat 2/26/11 Wed 3/2/11
10 Assigment 9 6 days Wed 3/2/11 Wed 3/9/11
11 Assigment 10 6 days Wed 3/9/11 Wed 3/16/11
12 Assigment 11 6 days Wed 3/16/11 Wed 3/23/11
13 Assigment 12 6 days Wed 3/23/11 Wed 3/30/11
14 Final Exam 0 days Wed 3/30/11 Wed 3/30/11
Jan 16, '11
Jan 23, '11
Jan 30, '11
Feb 6, '11
Feb 13, '11
Feb 20, '11
Feb 27, '11
Ready
New Tasks : Manually Scheduled
Desktop
6:45 PM
1/29/2011

From 2:40 to 3:30 – 50 minutes.

END

Planning and Estimating

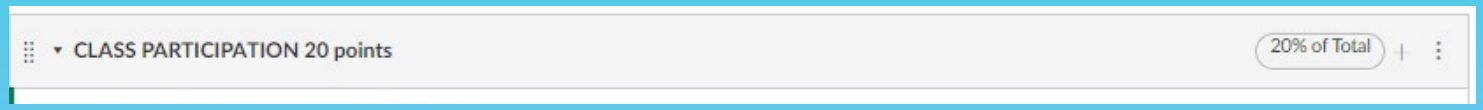
At 3:30 PM.

VH, next EXAM 2 Review

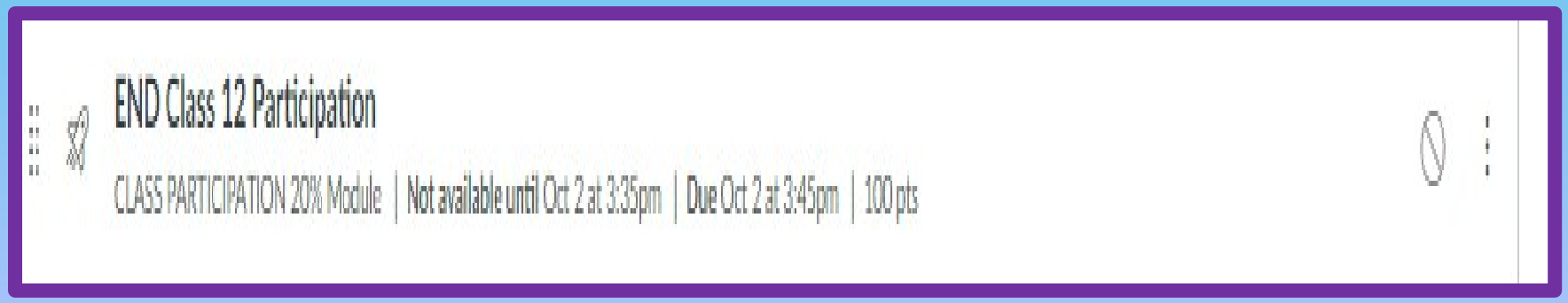
10.04.2023 (W 2:30 to 4) (13)		EXAM 2 REVIEW (CANVAS) (ZyBook)	Download ZyBook: Sections 6-9		
10.09.2023 (M 2:30 to 4) Optional (14)					Q & A Set 2 topics.
10.11.2023 (W 2:30 to 4) (15)					EXAM 2 (CANVAS) (ZyBook)

From 3:35 to 3:45 – 10 minutes.

10.02.2023 (M 2:30 to 4) (12)		Lecture 4: Estimating and Planning Tutorials 4 COCOMO and MS Project		Estimating and Planning Papers Summary (1 Page) CANVAS Assignment	



PASSWORD: CLASS 12



VH– Publish.

At 3:45.

End Class 12

VH, Download Attendance Report
Rename it:
10.02.2023 Attendance Report FINAL

VH, upload Class 12 to CANVAS.