

Project Management – Estimation

Week 11

Announcement

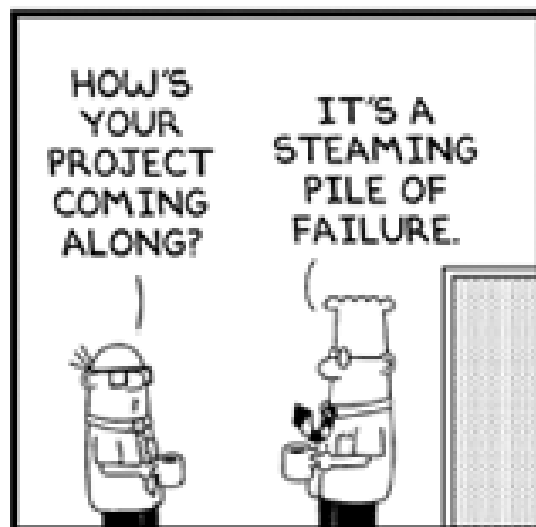
- Midterm 2
 - Wednesday, May. 4
 - Scope
 - Week 11 – Week 13
 - Short answer questions

Agenda (Lecture)

- Estimation

Agenda (Lab)

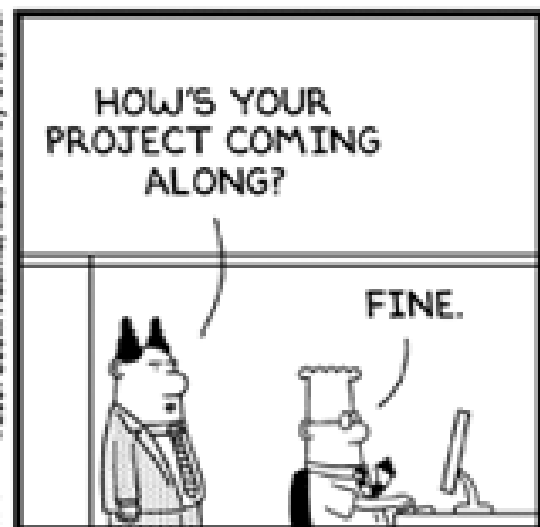
- Implement a software product based on your design documents
- Submit a weekly project progress report at the end of the Wednesday lab session



www.dilbert.com
scottadams@aol.com



3-15-97 © 2001 Scott Adams, Inc./Dist. by UFS, Inc.



© Scott Adams, Inc./Dist. by UFS, Inc.

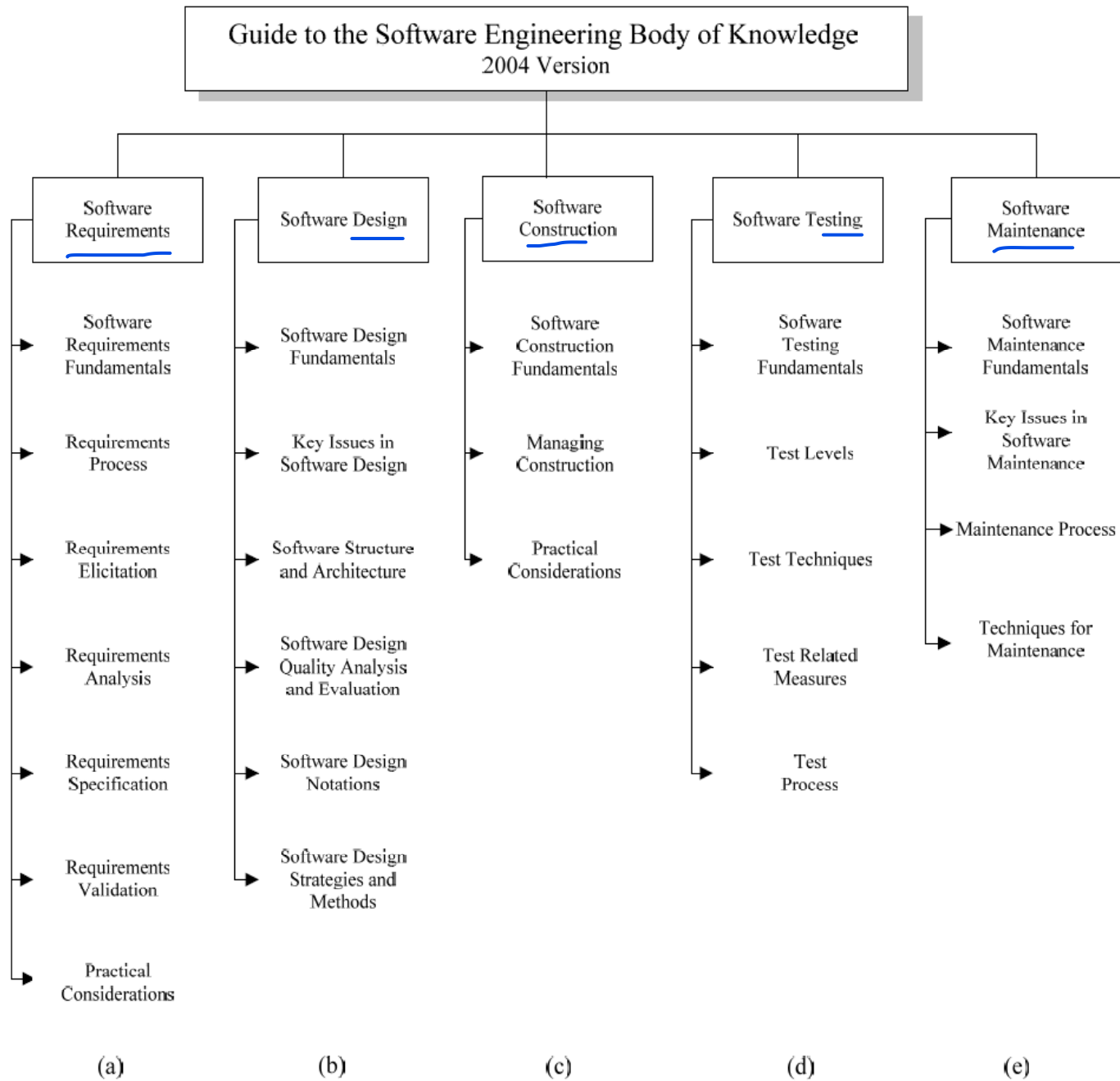


Figure 2 First five KAs

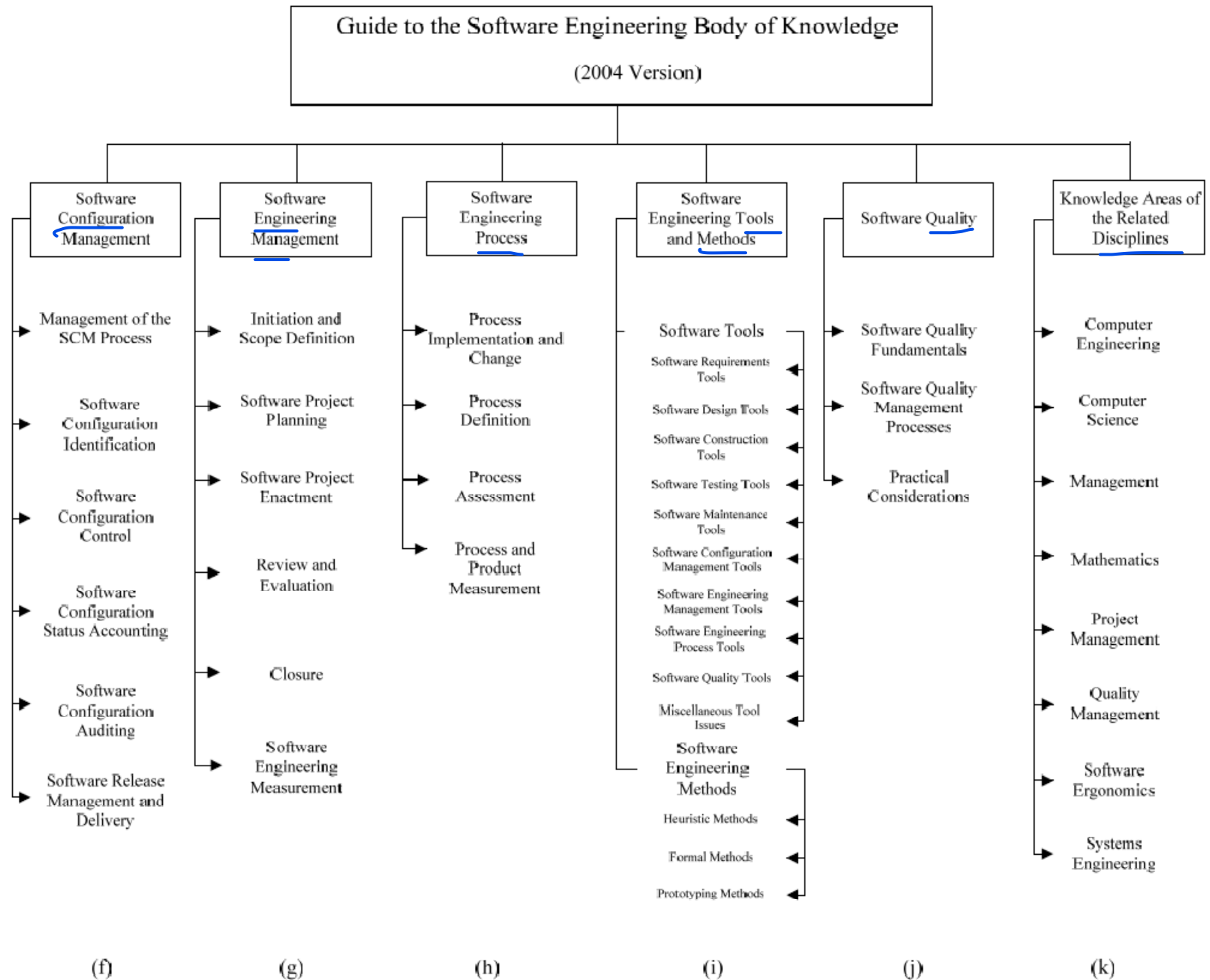


Figure 3 Last six KAs

Software Project Success Rate

Data on 280,000 projects completed in 2000 - *Standish Group Data*



<http://www.softwagemag.com/archive/2001feb/CollaborativeMgt.html>

Statements about Management

- “Software project management is an essential part of software engineering.”
- “Without proper planning, a software development project is doomed.”
- “Good management cannot guarantee project success. However, bad management usually result in project failure: The software is delivered late, costs more than originally estimated, and fails to its requirement.”

Project

- Organizations perform works: operations and projects
- Commonalities between operations and projects
 - Performed by people
 - Constrained by the limited resources
 - Planned, executed, and controlled
- Differences between operations and projects
 - Operations are on-going and repetitive
 - Projects are temporary and unique

Project Management

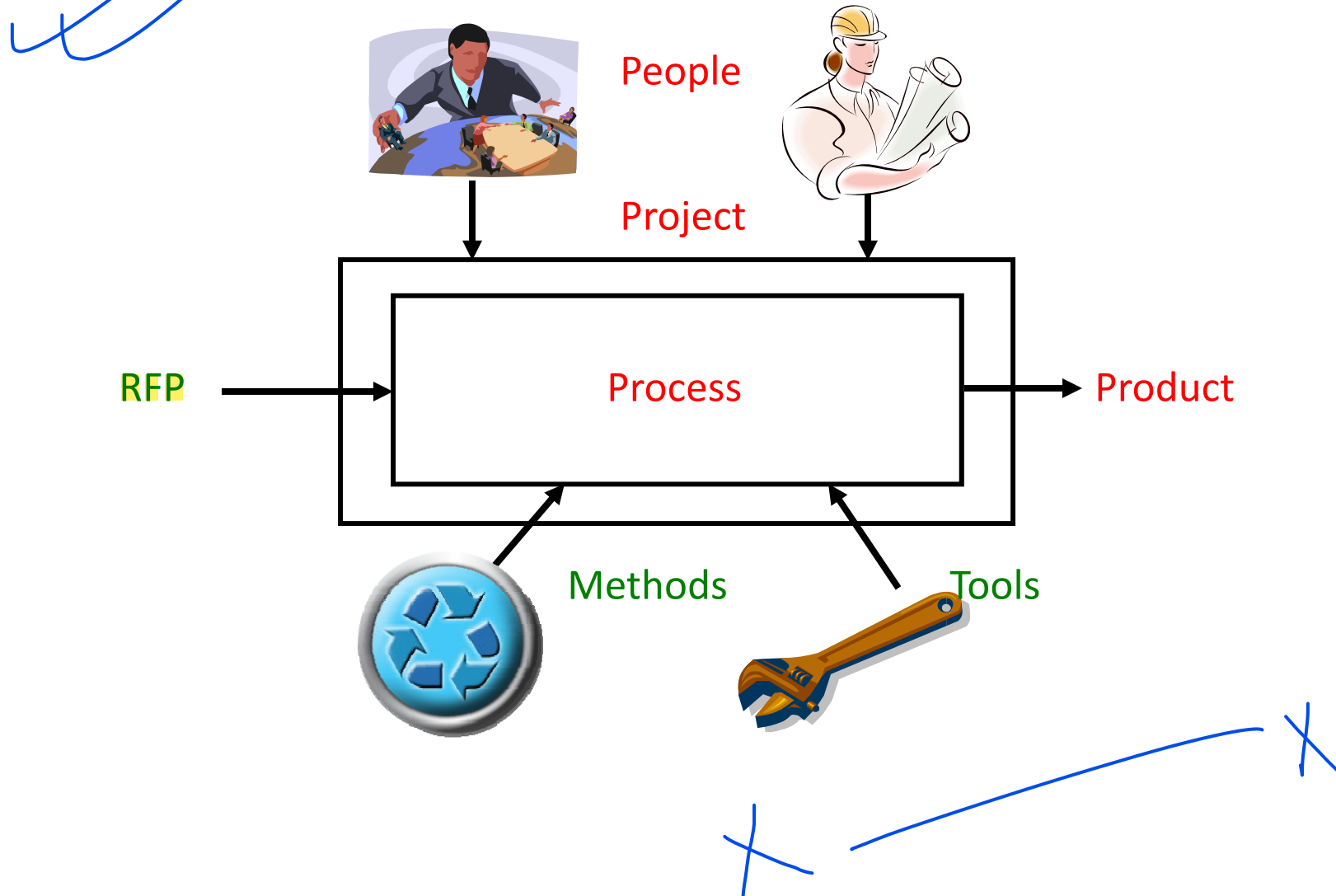
- Project Management Body Of Knowledge (PMBOK)
 - Project Management Institute
- www.csun.edu/~twang/380/Slides/pmbok.pdf

Software Project Management

- Software project management is especially difficult because
- IEEE Guide -- Adoption of PMI Standard A Guide to the Project Management Body of Knowledge -- IEEE Std 1490-1998
- ✓ • IEEE Standard for Software Project Management Plans -- IEEE Std 1058-1998
- Software project management : The Manager's View

up

Process/Project/Product/People




Metrics

- **Numerical measures** that quantify the degree to which software, a process or a project possesses a given attribute
- Metrics help the followings
 - Determining software **quality level**
 - **Estimating** project **schedules**
 - **Tracking** schedule process
 - Determining software **size** and **complexity**
 - Determining project **cost**
 - Process **improvement**

Software Metrics

- Without measure it is impossible to make a plan, detect problems, and improve a process and product
- A software engineer collects measure and develops metrics so that indicators will be obtained
- An indicator provides insight that enables the project manager or software engineers to adjust the process, the project, or the product to make things better

Software Metrics (cont'd)

- 
- The five essential, fundamental metrics:
 - Size (LOC, etc.)
 - Cost (in dollars)
 - Duration (in months)
 - Effort (in person-month)
 - Quality (number of faults detected)



Product Size Metrics



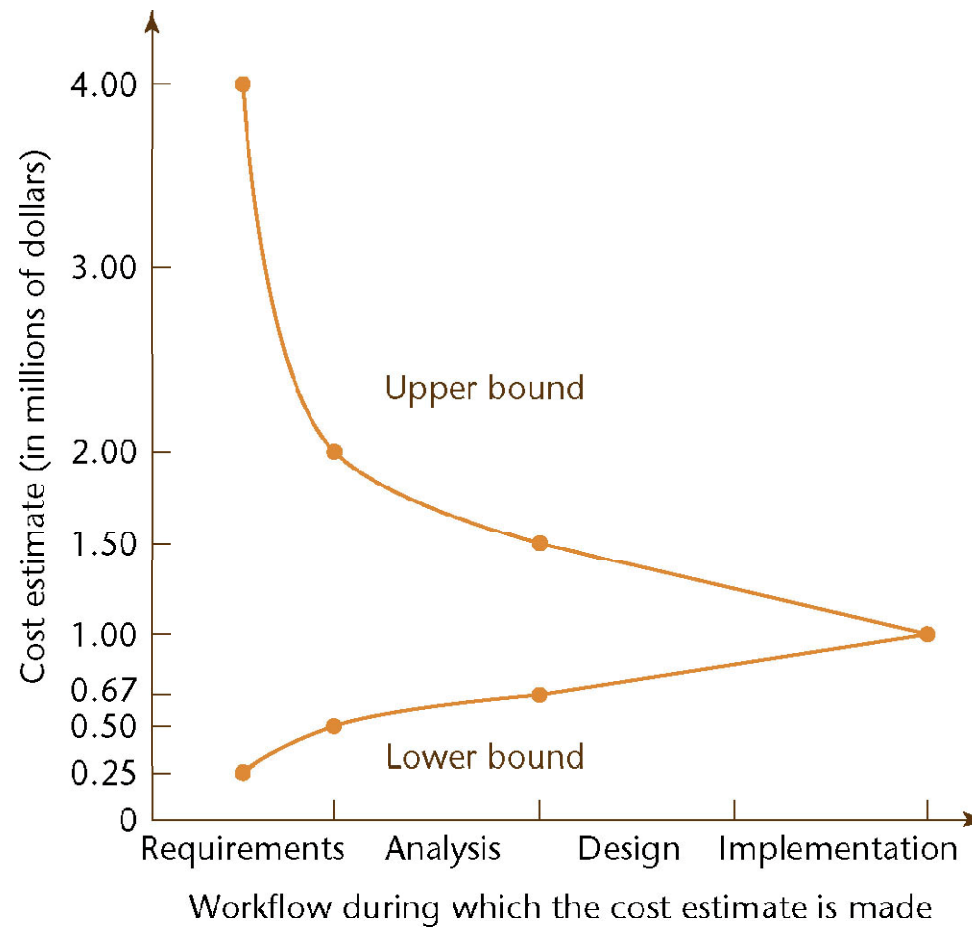
- Conventional metrics
 - Size-oriented metrics
 - Function-oriented metrics
 - Empirical estimation models
- Object-Oriented metrics
 - Number of scenario scripts
 - Number of key classes
 - Number of support classes
 - Average number of support classes per key classes
- User-Case oriented metrics

Product Size Metrics (cont'd)

- Web engineering product metrics
 - Number of static web pages
 - Number of dynamic web pages
 - Number of internal page links
 - Number of persistent page links




Estimate Uncertainty



- The accuracy of estimation increases as the process proceeds



Size Estimation

- The methods to achieve reliable size and cost estimates:
 - LOC-based estimation
 - FP-based estimation
 - Empirical estimation models
 - COCOMO
- 



LOC-based Estimation

- The problems of lines of code (LOC)
 - Different languages lead to different lengths of code
 - It is not clear how to count lines of code
 - A report, screen, or GUI generator can generate thousands of lines of code in minutes
 - Depending on the application, the complexity of code is different

LOC-based Estimation - Example

• Function	• Estimated LOC
– User interface	2,300
– 2-D geometric analysis	5,300
– 3-D geometric analysis	6,800
– Database management	3,500
– Graphic display facilities	4,950
– I/O control function	2,100
– Analysis function	8,400
• Total estimated LOC	<u>33,350</u>

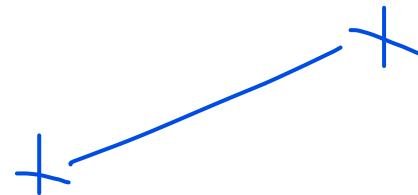
LOC-based Estimation - Exercise

- Average productivity based on historical data

[
– 620 LOC/pm
– \$8,000 per month
–> \$12.91/LOC

- If the estimated project is 33,200 LOC,
 - then the total estimated project cost is \$_____ and
 - the estimated effort is __ person-months

①



②

FP-based Estimation

- Based on FP metric for the size of a product
 - Based on the number of **inputs** (Inp), **outputs** (Out), **inquiries** (Inq), **master files** (Maf), **interfaces** (Inf)
 - **Step 1**: Classify each component of the product (Inp, Out, Inq, Maf, Inf) as **simple, average, or complex** (Figure 1)
 - Assign the appropriate number of function points
 - The sum of function pointers for each component **gives UFP** (unadjusted function points)

UFP

FP-based Estimation (cont'd)

- **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 (Figure 2)
 - **Add the 14 numbers:** This gives the **total degree of influence (DI)**
 - $TCF = 0.65 + 0.01 \times DI$
 - The technical complexity factor (TCF) lies between 0.65 and 1.35 ✓
- **Step 3.** The number of function points (FP) is then given by
 - $FP = UFP \times TCF$



FP-based Estimation (cont'd)

Component	Level of Complexity		
	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 1

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

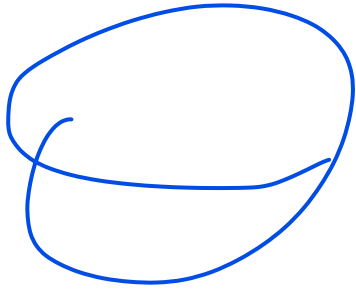
Figure 2

FP-based Estimation (cont'd)

- The same product was coded both in assembler and in ADA and the results compared

Thousands of
Delivered
Source
Instructions
KDSI

	Assembler Version	Ada Version
Source code size	70 <u>KDSI</u>	— 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



Exercise Problems

- A target product has 7 simple inputs, 2 average input, and 10 complex inputs. There are 56 average output, 8 simple inquiries, 12 average master files, and 17 complex interfaces. Determine the unadjusted function points (UFP).
- If the **total degree of influence** for the product of the question above is 49, determine the number of **function points**.

Average LOC Per One Function Point

Programming Languages	LOC/FP (average)
Assembly Language	320
C	128
COBOL	105
FORTRAN	106
Pascal	90
C++	64
Ada95	53
Visual Basic	32
Smalltalk	22
Powerbuilder	16
SQL	12



③

COCOMO

- COnstructive COst MOdel
- Empirical model
 - Metrics such as LOC and FP are used as input to a model for determining product cost and duration
- Well documented, and supported by public domain and commercial tools; Widely used and evaluated
- Has a long pedigree from its first instantiation in 1981
 - COCOMO I (81)
 - COCOMO II

COCOMO (cont'd)

- Based on water fall process model
- The vast majority of software would be developed from the scratch
- There are three forms of the COCOMO
 - Basic COCOMO (macro estimation) which gives an initial rough estimate of man months and development time
 - Intermediate COCOMO which gives a more detailed estimate for small to medium sized projects
 - Detailed COCOMO (micro estimation) which gives a more detailed estimate for large projects.

COCOMO (cont'd)

- $\text{Effort} = A * \text{Size}^B * M$
 - Where A is coefficient
 - The exponent B reflects the increased effort required as the size of the product increases
 - ✓ – The multiplier M is based on the project characteristics


Intermediate COCOMO

Organic mode (Simple)	Semi-detached mode (Moderate)	Embedded mode (Complex)
$MM_d = 3.2(KLOC)^{1.05}M$ $(NE = 3.2(KLOC)^{1.05})$	$MM_d = 3.0(KLOC)^{1.12}M$ $(NE = 3.0(KLOC)^{1.12})$	$MM_d = 2.8(KLOC)^{1.20}M$ $(NE = 2.8(KLOC)^{1.20})$

- **NE: Nominal effort** (a rough estimate of the development effort using two parameters)
- **MM_d : Man-month** for estimated development effort
- **M: 15 software development effort** multipliers
- KLOC: number of thousands of line of code

Intermediate COCOMO (cont'd)

- Step 1. Estimate the length of the product in KLOC
- Step 2. Estimate the product development mode
 - Simple (organic, straightforward)
 - Moderate (medium sized, semidetached)
 - Complex (embedded)
- Step 3. Compute the nominal effort
- Step 4. Multiply the nominal value by 15 software development cost multipliers
- Step 5. Estimate the calendar time (TDEV) in months required to complete a project



Cost Drivers	Rating					
	Very Low	Low	Nominal	High	Very High	Extra High
Product Attributes						
Required software reliability	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.						

Figure 5. Intermediate COCOMO software development effort multipliers

Intermediate COCOMO

– Example

- Example: Microprocessor-based communications processing software for electronic funds transfer network
- Step 1. Estimate the length of the product
 - 10,000 LOC (10 KLOC)
- Step 2. Estimate the product development mode
 - Complex (“embedded”) mode
- Step 3. Compute the nominal effort
 - Nominal effort = $2.8 * (10)^{1.20} = 44$ man-months

Intermediate COCOMO

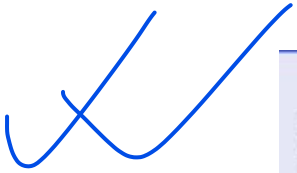
- Example (cont'd)

- Step 4. Multiply the nominal value by 15 software development cost multipliers (see table on the next slide)
 - Product of effort multipliers = 1.35
 - Estimated effort for project is therefore $1.35 * 44 = 59$ person (man)-months



Intermediate COCOMO

- Example (cont'd)



Cost Drivers	Situation	Rating	Effort Multiplier
Required software reliability	Serious financial consequences of software fault	High	1.15
Data base 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
Virtual machine volatility	Based on commercial microprocessor hardware	Nominal	1.00
Computer turnaround time	2 hour average turnaround time	Nominal	1.00
Analyst capabilities	Good senior analysts	High	0.86
Applications experience	3 years	Nominal	1.00
Programmer capability	Good senior programmers	High	0.86
Virtual machine experience	6 months	Low	1.10
Programming language experience	12 months	Nominal	1.00
Use of modern programming practices	Most techniques in use over 1 year	High	0.91
Use of software tools	At basic minicomputer tool level	Low	1.10
Required development schedule	9 months	Nominal	1.00

Results of the Intermediate COCOMO

- COCOMO has been validated with respect to broad samples (63)
- COCOMO was the most accurate estimation method of its time
- Major problem
 - If the estimate of the number of lines of codes of the target product is incorrect, then everything is incorrect

COCOMO II

- 1995 extension to 1981 COCOMO that incorporates
 - Object orientation, Modern life-cycle models, Rapid prototyping, Fourth-generation languages, COTS software
- COCOMO II is far more complex than the first version

Exercise Problem

- You are in charge of developing a 76-KLOC embedded product that is nominal except that the database size is rated very high and the use of software tools is low. Using Intermediate COCOMO, what is the estimated effort in person (man)-months?

