

# (Optional) Group Project Consultation

## During (Optional) Feedback Session on Friday

**1<sup>st</sup> Nov** Group 1 (3pm), 5 (3:15pm), 10 (3:30pm)

**8<sup>th</sup> Nov** Group 6 (3pm), 7 (3:15pm), 12 (3:30pm)

**15<sup>th</sup> Nov** Group 2 (3pm), 8 (3:15pm), 15 (3:30pm)

**29<sup>th</sup> Nov** Group 4 (3pm), 11 (3:15pm), 16 (3:30pm)

**6<sup>th</sup> Nov** Group 9 (3pm), 14 (3:15pm), 18 (3:30pm)

**13<sup>th</sup> Nov** Group 13 (3pm), 20 (3:15pm)

## During Computer Labs

**Any group**

**Give your feedback - Stop / Start /  
Continue**



# CO7095 – Software Measurement & Quality Assurance

Dr Fuxiang Chen

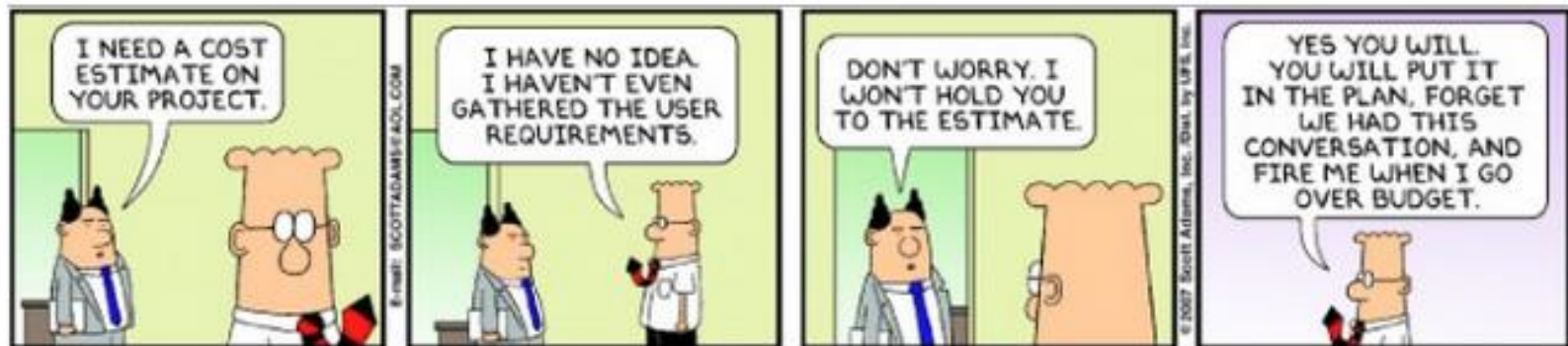
## Lecture 5

# Schedule & Cost Estimation II















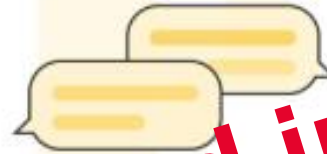




# How to play planning poker



1. Read out and discuss the story



3. Discuss discrepancies

2. Select and show your cards



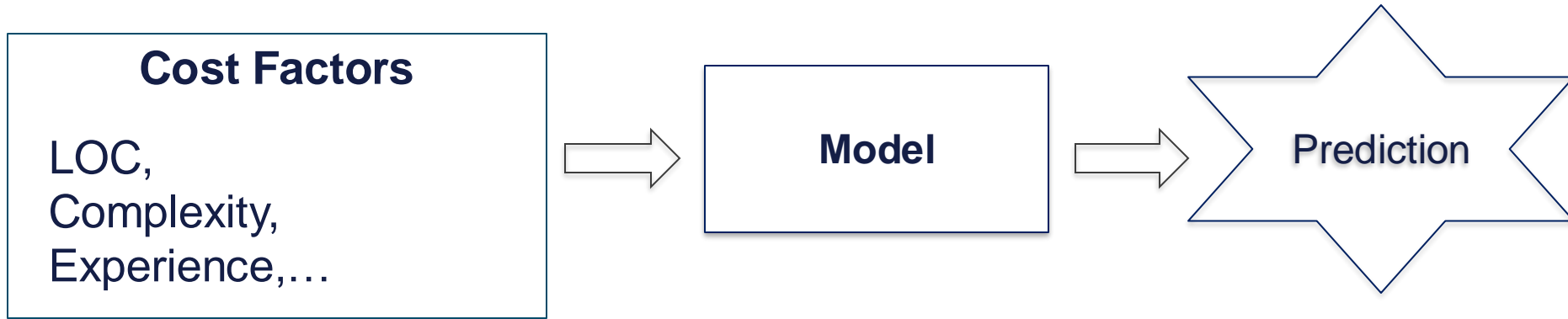
4. Repeat your vote if needed



**You have covered in last week's lab!**

# Model Based Software Effort/Cost Estimation

# Approach



# A Simple Ballpark Model

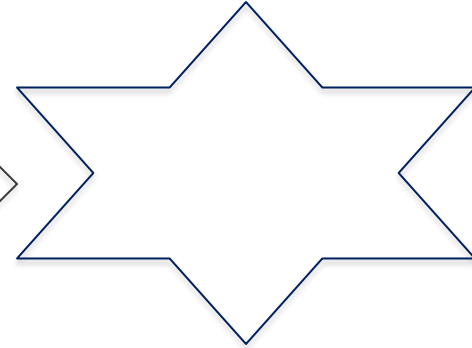
## Cost Factors

a: Cost per KLOC

s: Predicted size (KLOC)



**Model**





# A Simple Ballpark Model

## Cost Factors

a: Cost per KLOC

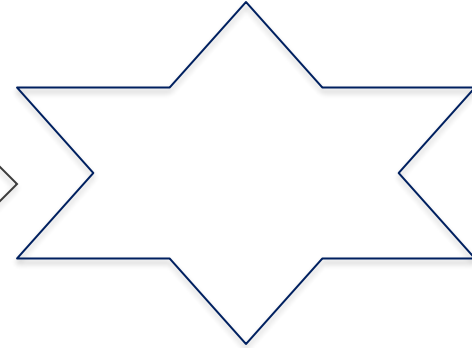
s: Predicted size (KLOC)



Effort/Cost



$$E = a \times s$$



# A Simple Ballpark Model

## Cost Factors

a: Cost per KLOC

s: Predicted size (KLOC)



Effort/Cost



$$E = a \times s$$



£3,000

*How much is the software project for an estimated 250,000 LOC,  
where cost per KLOC is £12?*

# Adding One-off Costs

## Cost Factors

a: Cost per KLOC

s: Predicted size (KLOC)

c: One-off costs



$$E = a \times s + c$$



£6,000

*How much is the software project for an estimated 250,000 LOC, where cost per KLOC is £12 with a start-up cost of £3,000?*

# Scale Factor

- Using size alone can skew models.
  - Relationship between size and cost may be **non-linear**.
- Other factors can play a role in costs.
  - Might apply disproportionately when the project is either small or large.
- Consider:
  - Management costs – may cost disproportionately more for larger projects.
  - Code reuse – could lead to disproportionate cost savings for larger projects.

# Adding Scale Factor

## Cost Factors

a: Cost per KLOC

s: Predicted size (KLOC)

c: One-off costs

b: Scale factor



$$E = a \times s^b + c$$



£15,000

*How much is the software project for an estimated 100,000 LOC, where cost per KLOC is £12 with a start-up cost of £3,000, and a scale factor of 1.5?*

# Parameter Fitting

# Choice of Parameters is Critical

- How do obtain parameter values?
  - What is the typical cost-per-LOC (a)?
  - What is the typical “scale factor” (b)?
  - What is the normal initial base-cost (c)?
- Can draw upon **historical data** from within organisation.
  - Software process should incorporate metrics and data collection.
  - Can be used to give indicators for these parameters.

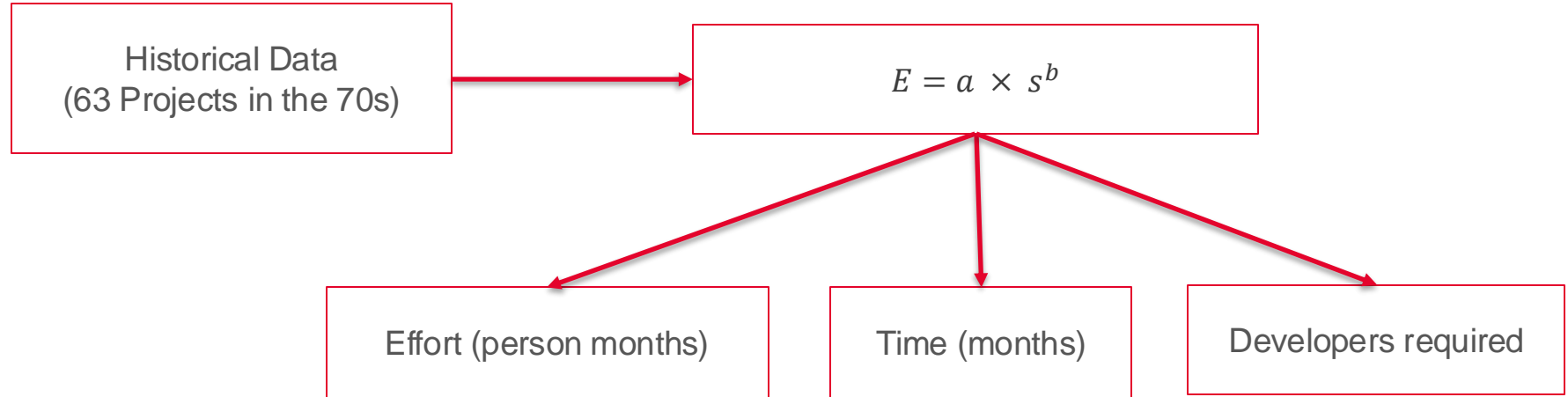
COCOMO



# COCOMO

- **CO**nstructive **CO**st **MO**del
- Developed by Barry Boehm in 1981

$E$  : Effort  
 $a$  : Cost per size-unit  
 $b$  : Scale factor  
 $s$  : Predicted size  
 $c$  : One-off costs



# Three Flavors

- **Basic**

- Project is in early stages.
- Subject to some very basic requirements capture.

- **Intermediate**

- Requirements have been captured to a reasonable degree of detail.

- **Detailed**

- Requirements have been fully captured.
- Design is complete.

# Basic Model

## - Project Types

### - Organic

- Small teams of experienced developers.
- Flexible requirements.

### - Semi-detached

- Medium teams.
- Mixed experience.
- Mix of flexible and inflexible requirements.

### - Embedded

- Tight requirements.
- Experienced team.



# Basic Model

Development Context	$a$	$b$	$c$	$d$
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

$$E = a \times s^b$$

**Person Months**

$$D = c \times E^d$$

**Months**

$$P = E \div D$$

**People**

# Basic Model (Example)

Development Context	$a$	$b$	$c$	$d$
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

$$E = a \times s^b$$

**Person Months** 10.289 Person Months

Given Development  
Context: Organic, and  
4KLOC

$$D = c \times E^d$$

**Months** 6.06237 Months

What is the Effort,  
Development Time and  
number of people  
required?

$$P = E \div D$$

**People** 2 People

# Intermediate Model: Adding Cost Drivers

<b>Product Attributes</b>	Required Software Reliability
	Size of Application Database
	Complexity of The Product
<b>Hardware Attributes</b>	Runtime Performance Constraints
	Memory Constraints
	Volatility of the virtual machine environment
	Required turnabout time
<b>Personnel Attributes</b>	Analyst capability
	Applications experience
	Software engineer capability
	Virtual machine experience
<b>Project Attributes</b>	Programming language experience
	Application of software engineering methods
	Use of software tools
	Required development schedule

Given a rating on an ordinal scale:

1. Very Low
2. Low
3. Nominal
4. High
5. Very High
6. Extra High

# Calculating the Effort Adjustment Factor

COST DRIVERS	VERY LOW	LOW	NOMINAL	HIGH	VERY HIGH	EXTRA HIGH
<b>Product Attributes</b>						
Required Software Reliability	0.75	0.88	1.00	1.15	1.40	
Size of Application Database		0.94	1.00	1.08	1.16	
Complexity of The Product	0.70	0.85	1.00	1.15	1.30	1.65
<b>Hardware Attributes</b>						
Runtime Performance Constraints			1.00	1.11	1.30	1.66
Memory Constraints			1.00	1.06	1.21	1.56
Volatility of the virtual machine environment		0.87	1.00	1.15	1.30	
Required turnabout time		0.94	1.00	1.07	1.15	
<b>Personnel Attributes</b>						
Analyst capability	1.46	1.19	1.00	0.86	0.71	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Software engineer 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		
<b>Project Attributes</b>						
Application of software engineering methods	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	

$$E = a \times s^b \times EAF$$

**Effort Adjustment Factor**  
(product of the weightings  
attributed to all the cost  
drivers)

# Calculating the Effort Adjustment Factor (Example)

COST DRIVERS	VERY LOW	LOW	NOMINAL	HIGH	VERY HIGH	EXTRA HIGH
<b>Product Attributes</b>						
Required Software Reliability	0.75	0.88	1.00	1.15	1.40	
Size of Application Database		0.94	1.00	1.08	1.16	
Complexity of The Product	0.70	0.85	1.00	1.15	1.30	1.65
<b>Hardware Attributes</b>						
Runtime Performance Constraints			1.00	1.11	1.30	1.66
Memory Constraints			1.00	1.06	1.21	1.56
Volatility of the virtual machine environment		0.87	1.00	1.15	1.30	
Required turnabout time		0.94	1.00	1.07	1.15	
<b>Personnel Attributes</b>						
Analyst capability	1.46	1.19	1.00	0.86	0.71	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Software engineer 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		
<b>Project Attributes</b>						
Application of software engineering methods	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	

$$E = a \times s^b \times EAF$$

**Effort Adjustment Factor**

For a given project was estimated with a size of 300 KLOC.

Calculate the EAF by considering the developer having very high application experience and very low experience in programming.



# Calculating the Effort Adjustment Factor (Example)

COST DRIVERS	VERY LOW	LOW	NOMINAL	HIGH	VERY HIGH	EXTRA HIGH
<b>Product Attributes</b>						
Required Software Reliability	0.75	0.88	1.00	1.15	1.40	
Size of Application Database		0.94	1.00	1.08	1.16	
Complexity of The Product	0.70	0.85	1.00	1.15	1.30	1.65
<b>Hardware Attributes</b>						
Runtime Performance Constraints			1.00	1.11	1.30	1.66
Memory Constraints			1.00	1.06	1.21	1.56
Volatility of the virtual machine environment		0.87	1.00	1.15	1.30	
Required turnabout time		0.94	1.00	1.07	1.15	
<b>Personnel Attributes</b>						
Analyst capability	1.46	1.19	1.00	0.86	0.71	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Software engineer 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		
<b>Project Attributes</b>						
Application of software engineering methods	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	

$$E = a \times s^b \times EAF$$

**Effort Adjustment Factor**

Estimated size of the project is: 300 KLOC

# Calculating the Effort Adjustment Factor (Example)

COST DRIVERS	VERY LOW	LOW	NOMINAL	HIGH	VERY HIGH	EXTRA HIGH
<b>Product Attributes</b>						
Required Software Reliability	0.75	0.88	1.00	1.15	1.40	
Size of Application Database		0.94	1.00	1.08	1.16	
Complexity of The Product	0.70	0.85	1.00	1.15	1.30	1.65
<b>Hardware Attributes</b>						
Runtime Performance Constraints			1.00	1.11	1.30	1.66
Memory Constraints			1.00	1.06	1.21	1.56
Volatility of the virtual machine environment		0.87	1.00	1.15	1.30	
Required turnabout time		0.94	1.00	1.07	1.15	
<b>Personnel Attributes</b>						
Analyst capability	1.46	1.19	1.00	0.86	0.71	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Software engineer 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		
<b>Project Attributes</b>						
Application of software engineering methods	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	

$$E = a \times s^b \times EAF$$

**Effort Adjustment Factor**

Estimated size of the project is: 300 KLOC

Developer having very high application experience: 0.82 (from table)

# Calculating the Effort Adjustment Factor (Example)

COST DRIVERS	VERY LOW	LOW	NOMINAL	HIGH	VERY HIGH	EXTRA HIGH
<b>Product Attributes</b>						
Required Software Reliability	0.75	0.88	1.00	1.15	1.40	
Size of Application Database		0.94	1.00	1.08	1.16	
Complexity of The Product	0.70	0.85	1.00	1.15	1.30	1.65
<b>Hardware Attributes</b>						
Runtime Performance Constraints			1.00	1.11	1.30	1.66
Memory Constraints			1.00	1.06	1.21	1.56
Volatility of the virtual machine environment		0.87	1.00	1.15	1.30	
Required turnabout time		0.94	1.00	1.07	1.15	
<b>Personnel Attributes</b>						
Analyst capability	1.46	1.19	1.00	0.86	0.71	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Software engineer 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		
<b>Project Attributes</b>						
Application of software engineering methods	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	

$$E = a \times s^b \times EAF$$

**Effort Adjustment Factor**

Estimated size of the project is: 300 KLOC

Developer having very high application experience: 0.82 (from table)

Developer having very low experience in programming: 1.14 (from table)

# Calculating the Effort Adjustment Factor (Example)

COST DRIVERS	VERY LOW	LOW	NOMINAL	HIGH	VERY HIGH	EXTRA HIGH
<b>Product Attributes</b>						
Required Software Reliability	0.75	0.88	1.00	1.15	1.40	
Size of Application Database		0.94	1.00	1.08	1.16	
Complexity of The Product	0.70	0.85	1.00	1.15	1.30	1.65
<b>Hardware Attributes</b>						
Runtime Performance Constraints			1.00	1.11	1.30	1.66
Memory Constraints			1.00	1.06	1.21	1.56
Volatility of the virtual machine environment		0.87	1.00	1.15	1.30	
Required turnabout time		0.94	1.00	1.07	1.15	
<b>Personnel Attributes</b>						
Analyst capability	1.46	1.19	1.00	0.86	0.71	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Software engineer 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		
<b>Project Attributes</b>						
Application of software engineering methods	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	

$$E = a \times s^b \times EAF$$

**Effort Adjustment Factor**

Estimated size of the project is: 300 KLOC

Developer having very high application experience: 0.82 (from table)

Developer having very low experience in programming: 1.14 (from table)

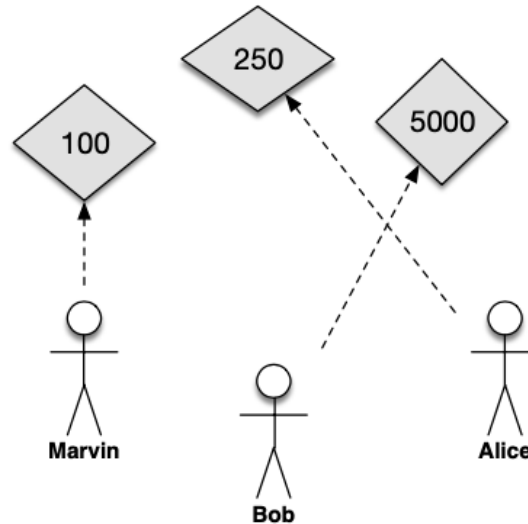
$$EAF = 0.82 \times 1.14 = 0.9348$$

*End of*  
Schedule & Cost  
Estimation II

# Measuring Developer's Productivity

# Example: Measuring Developer's Productivity

- Measure by number of lines of code written



Is this measurement valid?  
If not, why not?

Designed entire system. Wrote  
highly efficient algorithmic core.

Copied and pasted license text into every source file.

# Measuring Software Size & Complexity



# Why are Size and Complexity Relevant?

- Before development...
  - *How much programming effort will module X require?*
  - *What will be the estimated cost of the final product?*
  - Metrics are based upon requirements / specification (“**black box**”)
- After development...
  - *How much effort will maintenance require?*
  - *Where should we direct testing effort?*
  - *How much effort did development require?*
  - Metrics are based upon source code (“**white box**”)

“Black-box” approaches to  
measuring size and  
complexity

# We have already discussed one such measure

- Can you remember what it was?
- Story points
  - Scale is not rooted in any development artefacts.
  - Approximate units of “effort”.
  - Zero points = zero effort.
  - Serve to compare, prioritise and plan development of user stories.

**You have covered in the lab!**



# Function Point Count

- Developed by Albrecht at IBM in 1979.
- Characterise size & complexity in terms of interaction components:
  - **External inputs** – e.g. provided by user
  - **External outputs** – e.g. via GUI to user
  - **External inquiries** – inputs solicited from user, e.g. via dialog box
  - **Internal files** – data stored to files internally
  - **External files** – data stored externally, e.g., to databases
- And each category is given a measure of “difficulty”:
  - 1. **Simple**
  - 2. **Average**
  - 3. **Difficult**


# Computing the Unadjusted Function Point Count (UFC)

Category	Simple	Average	Difficult
1: External inputs	3	4	6
2: External outputs	4	5	7
3: External inquiries	3	4	6
4: Internal files	7	10	15
5: External files	5	7	10

Predefined Weights

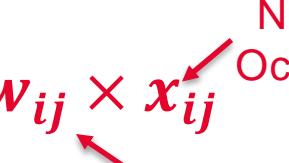
# Computing the Unadjusted Function Point Count (UFC)

Category	Simple	Average	Difficult	
1: External inputs	3×4	4×3	6×1	12 + 12 + 6 = 30
2: External outputs	4×0	5×2	7×1	0 + 10 + 7 = 17
3: External inquiries	3×7	4×0	6×0	21 + 0 + 0 = 21
4: Internal files	7×3	10×0	15×0	21 + 0 + 0 = 21
5: External files	5×0	7×5	10×2	0 + 35 + 20 = 55


  
 Number of Occurrences

144

$$UFC = \sum_{i=1}^5 \sum_{j=1}^3 w_{ij} \times x_{ij}$$


  
 The weights

# Incorporating the Technical Complexity Factor

- The Technical Complexity Factor (TCF) quantifies complexity that is not considered by the UFC.
- 14 “technical factors” are rated on scale from 0 (no influence) to 5 (critically influential)
- $TCF = 0.65 + 0.01 \times \sum_{i=1}^{14} F_i$
- $FPC = UFC \times TCF$

Technical  
factor

General System Characteristic	
1	How many <b>data communication</b> facilities are there to aid in the transfer or exchange of information with the application or system?
2	How are distributed data and processing functions handled? ( <b>Distributed data processing</b> )
3	Did the user require response time or throughput? ( <b>Performance</b> )
4	How heavily used is the current hardware platform where the application will be executed? ( <b>Heavily used configuration</b> )
5	How frequently are transactions executed? ( <b>Transaction rate</b> )
6	What % of the information is entered On-Line? ( <b>On-line data entry</b> )
7	Was the application designed for <b>end-user efficiency</b> ?
8	How many ILF's are updated by On-Line transaction? ( <b>Online update</b> )
9	Does the application have <b>complex</b> logical or mathematical <b>processing</b> ?
10	Was the application developed to meet <b>reusability</b> needs?
11	How difficult is conversion and <b>installation</b> ?
12	How effective and/or automated are start-up, back up, and recovery procedures? ( <b>Operational ease</b> )
13	Was the application developed to be installed at <b>multiple sites</b> ?
14	Was the application developed to <b>facilitate change</b> ?

“White-box” approaches  
to measuring size and  
complexity



# Number of Lines of Code

- Number of lines in a source code file (or group of files)
- Widely derided as a useful metric.

Blank Lines?

Comments?

Not all “lines” are equal.

**Why  
Objection?**

Strongly affected by formatting conventions.

Highly language-specific.

Ignores logical / architectural complexity.

# Derided .... But often useful

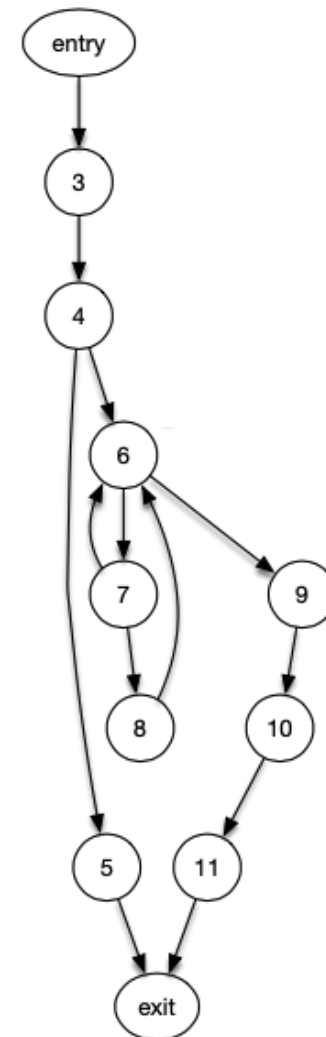
- Easy to compute
  - In its basic form, it can be computed without needing to parse source code
- Easy to understand and interpret
  - If the definition of a “line” is clearly stated
- Often sufficient for an approximate measure of size
  - Widely used (perhaps the most widely used) metric



# Cyclomatic Complexity

- Developed by Thomas McCabe in 1976.
- **Goal:** Compute logical complexity.
- Number of independent paths through the code.
- Based on control flow graph.
- $|\text{Edges}| - |\text{Nodes}| + 2 \times P$
- P is number of procedures (usually 1)

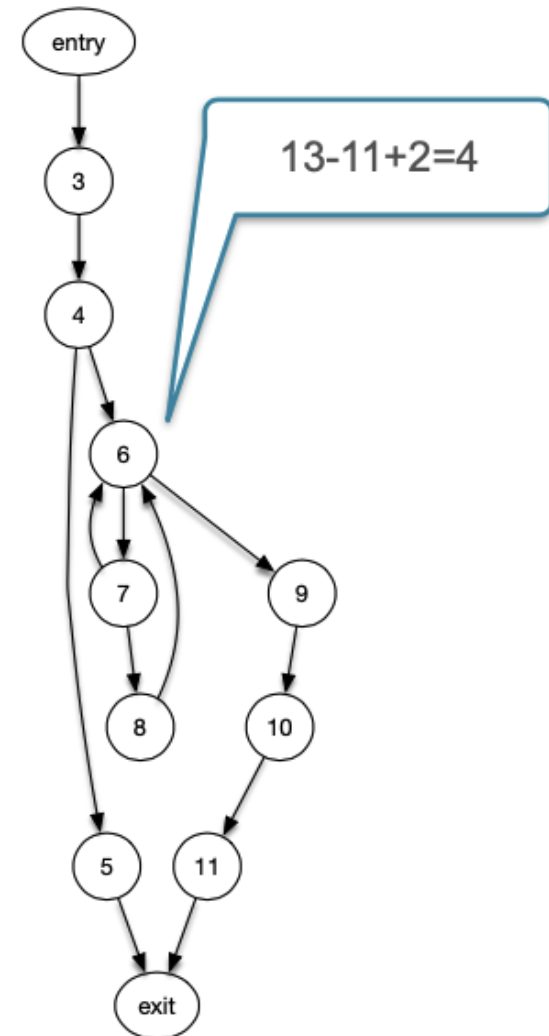
```
void iqsort0(int *a, int n)
{
    int i, j;
    if (n <= 1)
        return;
    for (i = 1, j = 0; i < n; i++)
        if (a[i] < a[0])
            swap(++j, i, a);
    swap(0, j, a);
    iqsort0(a, j);
    iqsort0(a+j+1, n-j-1);
}
```



# Cyclomatic Complexity

- Developed by Thomas McCabe in 1976.
- **Goal:** Compute logical complexity.
- Number of independent paths through the code.
- Based on control flow graph.
- **$|\text{Edges}| - |\text{Nodes}| + 2 \times P$**
- P is number of procedures (usually 1)

```
void iqsort0(int *a, int n)
{
    int i, j;
    if (n <= 1)
        return;
    for (i = 1, j = 0; i < n; i++)
        if (a[i] < a[0])
            swap(++j, i, a);
    swap(0, j, a);
    iqsort0(a, j);
    iqsort0(a+j+1, n-j-1);
}
```



# Summary

- COCOMO is based on historical software development data (historical projects)
- We have covered a variety of metrics.
  - Aim to measure various facets of software and software development process.