Software Cost Estimation
Part 2

Lecture 28

Function points and LOC

- FPs can be used to estimate LOC depending on the average number of LOC per FP for a given language
  - LOC = AVC * number of function points
  - AVC is a language-dependent factor varying from approximately 300 for assemble language to 12-40 for a 4GL

Relation Between FP & LOC

Programming Language	LOC/FP (average)
Assembly language	320
C	128
COBOL	106
FORTRAN	106
Pascal	90
C++	64
Ada	53
Visual Basic	32
Smalltalk	22
Power Builder (code generator)	16
SQL	12

Function Points & Normalisation

- Function points are used to normalise measures (same as for LOC) for:
  - S/w productivity
  - Quality

- Error (bugs) per FP (discovered at programming)
- Defects per FP (discovered after programming)
- $ per FP
- Pages of documentation per FP
- FP per person-month

Expected Software Size

- Based on three-point
- Compute Expected Software Size (S) as weighted average of:
  - Optimistic estimate: S(opt)
  - Most likely estimate: S(ml)
  - Pessimistic estimate: S(pess)
- $S = \{ S(opt) + 4\, S(ml) + S(pess) \} / 6$
  - Beta probability distribution

Example 1: LOC Approach

- A system is composed of 7 subsystems as below.
- Given for each subsystem the size in LOC and the
  2 metrics: productivity LOC/pm (pm: person month) ,Cost $/LOC
- Calculate the system total cost in $ and effort in months .

Functions	estimated LOC	LOC/pm	$/LOC
UICF	2340	315	14
2DGA	5380	220	20
3DGA	6800	220	20
DSM	3350	240	18
CGDF	4950	200	22
PCF	2140	140	28
DAM	8400	300	18

Example 1: LOC Approach

Functions	estimated LOC	LOC/pm	$/LOC	Cost	Effort (months)
UICF	2340	315	14	32,000	7.4
2DGA	5380	220	20	107,000	24.4
3DGA	6800	220	20	136,000	30.9
DSM	3350	240	18	60,000	13.9
CGDF	4950	200	22	109,000	24.7
PCF	2140	140	28	60,000	15.2
DAM	8400	300	18	151,000	28.0
Totals	33,360			655,000	145.0

Example 2: LOC Approach

Assuming

- Estimated project LOC = 33200
- Organisational productivity (similar project type) = 620 LOC/p-m
- Burdened labour rate = 8000 $/p-m

Then

- Effort = 33200/620 = (53.6) = 54 p-m
- Cost per LOC = 8000/620 = (12.9) = 13 $/LOC
- Project total Cost = 8000 * 54 = 432000 $

Example 3: FP Approach

	A	B	C	D	E	F	G
1	Info Domain	Optimistic	Likely	Pessim.	Est Count	Weight	FP count
2	# of inputs	22	26	30	26	4	104
3	# of outputs	16	18	20	18	5	90
4	# of inquiries	16	21	26	21	4	84
5	# of files	4	5	6	5	10	50
6	# of external interf	1	2	3	2	7	14
7	UFC: Unadjusted Function Count						342
8			Complexity adjustment factor				1.17
9						FP	400

Example 3: FP Approach (cont.)
Complexity Factor

Complexity factor: Fi	value=0	value=1	value=2	value=3	value=4	value=5	Fi
Backup and recovery	0	0	0	0	1	0	4
Data communication	0	0	1	0	0	0	2
Distributed processing functions	0	0	0	0	0	0	0
Is performance critical?	0	0	0	0	1	0	4
Existing operating environment	0	0	0	1	0	0	3
On-line data entry	0	0	0	0	1	0	4
Input transaction built over multiple screens	0	0	0	0	0	1	5
Master files updated on-line	0	0	0	1	0	0	3
Complexity of inputs, outputs, files, inquiries	0	0	0	0	0	1	5
Complexity of processing	0	0	0	0	0	1	5
Code design for re-use	0	0	0	0	1	0	4
Are conversion/installation included in design?	0	0	0	1	0	0	3
Multiple installations	0	0	0	0	0	1	5
Application designed to facilitate change by the user	0	0	0	0	0	1	5
						Sigma (F)	52
Complexity adjustment factor	0.65 + 0.01 * Sigma (F) =			1.17			

Example 3: FP Approach (cont.)

Assuming $\sum_i F_i = 52$

$$FP = UFC * \left[0.65 + 0.01 * \sum_i F_i \right]$$

FP = 342 * 1.17 = 400

Complexity adjustment factor = 1.17

Example 4: FP Approach (cont.)

Assuming

- Estimated FP = 401
- Organisation average productivity (similar project type) = 6.5 FP/p-m (person-month)
- Burdened labour rate = 8000 $/p-m

Then

- Estimated effort = 401/6.5 = (61.65) = 62 p-m
- Cost per FP = 8000/6.5 = 1231 $/FP
- Project cost = 8000 * 62 = 496000 $

Object Points (for 4GLs)

- Object points are an alternative function-related measure to function points when 4Gls or similar languages are used for development
- Object points are NOT the same as object classes
- The number of object points in a program is a weighted estimate of
  - The number of separate screens that are displayed
  - The number of reports that are produced by the system
  - The number of 3GL modules that must be developed to supplement the 4GL code
  - C:\Software Eng\Cocomo\Software Measurement Page, COCOMO II, object points.htm

Object Points – Weighting

Object Type	Simple	Meduim	Difficult
Screen	1	2	3
Report	2	5	8
Each 3GL module	10	10	10

Object Points – Weighting (cont.)

- ***srvr***: number of server data tables used with screen/report
- **clnt:** number of client data tables used with screen/report

For Screens				For Reports			
Number of Views contained	# and source of data tables			Number of Sections contained	# and source of data tables		
	Total < 4 (< 2 srvr < 3 clnt)	Total < 8 (2/3 srvr 3-5 clnt)	Total 8+ (> 3 srvr > 5 clnt)		Total < 4 (< 2 srvr < 3 clnt)	Total < 8 (2/3 srvr 3-5 clnt)	Total 8+ (> 3 srvr > 5 clnt)
< 3	simple	simple	medium	0 or 1	simple	simple	medium
3 - 7	simple	medium	difficult	2 or 3	simple	medium	difficult
> 8	medium	difficult	difficult	4 +	medium	difficult	difficult

Object Point Estimation

- Object points are easier to estimate from a specification than function points
  - simply concerned with screens, reports and 3GL modules
- At an early point in the development process:
  - Object points can be easily estimated
  - It is very difficult to estimate the number of lines of code in a system

Productivity Estimates

- LOC productivity
  - Real-time embedded systems, 40-160 LOC/P-month
  - Systems programs , 150-400 LOC/P-month
  - Commercial applications, 200-800 LOC/P-month
- Object points productivity

Developer's experience and Capability / ICASE maturity and capability	Very low	Low	Nominal	High	Very high
PROD: Productivity Object-point per person-month	4	7	13	25	50

Object Point Effort Estimation

- Effort in p-m = NOP / PROD
  - NOP = number of OP of the system
  - Example: An application contains 840 OP (NOP=840) & Productivity is very high (= 50)
  - Then, Effort = 840/50 = (16.8) = 17 p-m

Adjustment for % of Reuse

- Adjusted NOP = NOP * (1 - % reuse / 100)
- Example:
  - An application contains 840 OP, of which 20% can be supplied by existing components.

Adjusted NOP = 840 * (1 – 20/100) = 672 OP

Adjusted effort = 672/50 = (13.4) = 14 p-m

Factors affecting productivity

Factor	Description
Application domain experience	Knowledge of the application domain is essential for effective software development. Engineers whœalready understand a domain are likely to be the most productive.
Process quality	The development process used can have a significant effect on productivity. This is covered in Chapter 31.
Project size	The larger a project, the more time required for team communications. Less time is available for development so individual productivity is reduced.
Technology support	Good support technology such as CASE tools, supportive configuration management systems, etc. can improve productivity.
Working environment	As discussed in Chapter 28, a quiet working environment with private work areas contributes to improved productivity.

Quality and Productivity

- All metrics based on **volume/unit** time are flawed because they do not take quality into account
- Productivity may generally be increased at the cost of quality
- If change is constant, then an approach based on *counting lines of code* (LOC) is not meaningful

Estimation techniques

- There is no simple way to make an accurate estimate of the effort required to develop a software system:
  - Initial estimates may be based on inadequate information in a user requirements definition
  - The software may run on unfamiliar computers or use new technology
  - The people in the project may be unknown
- Project cost estimates may be self-fulfilling
  - The estimate defines the budget and the product is adjusted to meet the budget

Estimation techniques

- Algorithmic cost modelling
- Expert judgement
- Estimation by analogy
- Parkinson's Law
- Pricing to win

Algorithmic code modelling

- A formula – empirical relation:
  - based on historical cost information and which is generally based on the size of the software
- The formulae used in a formal model arise from the analysis of historical data.

Expert Judgement

- One or more experts in both software development and the application domain use their experience to predict software costs. Process iterates until some consensus is reached.
- Advantages: Relatively cheap estimation method. Can be accurate if experts have direct experience of similar systems
- Disadvantages: Very inaccurate if there are no experts!

Estimation by Analogy

- Experience-based Estimates
- The cost of a project is computed by comparing
  the project to a similar project in the same application domain
- Advantages: Accurate if project data available
- Disadvantages: Impossible if no comparable project has been tackled. Needs systematically maintained cost database

Estimation by Analogy : Problems

- However, new methods and technologies may make estimating based on experience inaccurate:
  - Object oriented rather than function-oriented development
  - Client-server systems rather than mainframe systems
  - Off the shelf components
  - Component-based software engineering
  - CASE tools and program generators

Parkinson's Law

- "The project costs whatever resources are available"

  (Resources are defined by the software house)

- Advantages: No overspend
- Disadvantages: System is usually unfinished
  - The work is contracted to fit the budget available: by reducing functionality, quality

Pricing to Win

- The project costs whatever the customer budget is.
- Advantages: You get the contract
- Disadvantages:
  - The probability that the customer gets the system he/she wants is small.
  - Costs do not accurately reflect the work required

Pricing to Win

- This approach may seem unethical and unbusiness like
- However, when detailed information is lacking it may be the only appropriate strategy
- The project cost is agreed on the basis of an outline proposal and the development is constrained by that cost
- A detailed specification may be negotiated or an evolutionary approach used for system development

Top-down and Bottom-up Estimation

- Top-down
  - Start at the system level and assess the overall system functionality
- Bottom-up
  - Start at the component level and estimate the effort required for each component. Add these efforts to reach a final estimate

Top-down Estimation

- Usable *without* knowledge of the system architecture and the components that might be part of the system
- Takes into account costs such as integration, configuration management and documentation
- Can underestimate the cost of solving difficult low-level technical problems

Bottom-up estimation

- Usable when
  - the architecture of the system is known and
  - components identified
- Accurate method if the system has been designed in detail
- May underestimate costs of system level activities such as integration and documentation

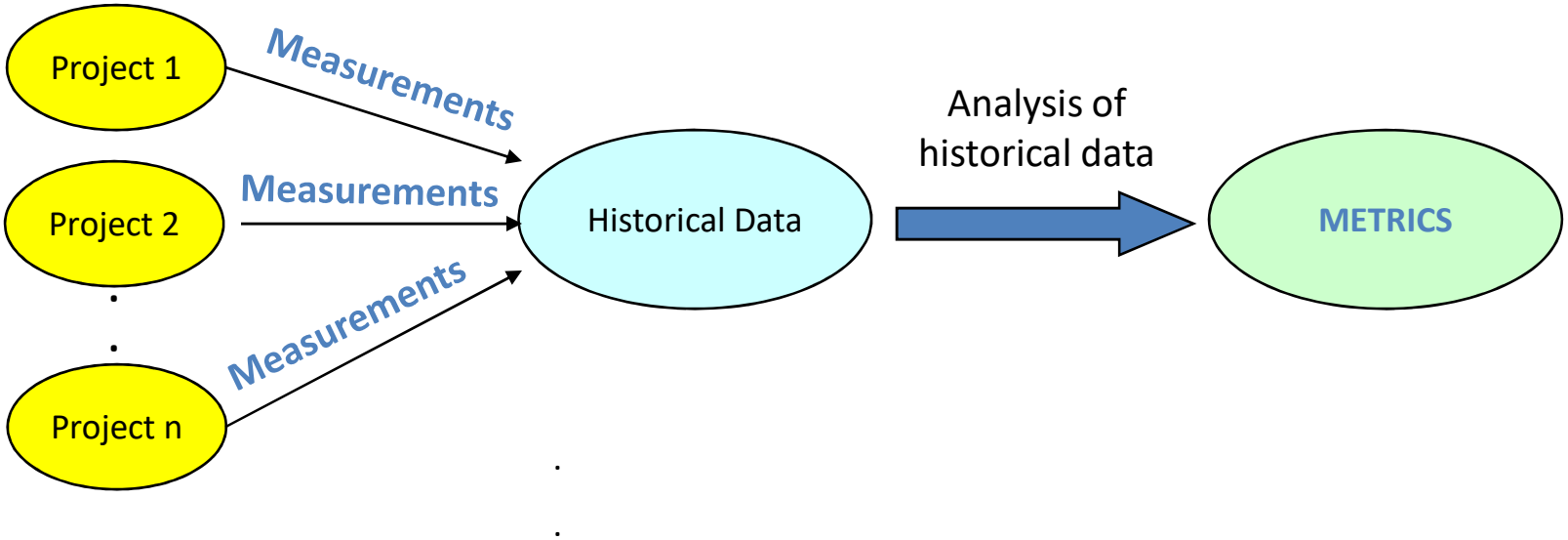
Estimation Methods

- S/W project estimation should be based on several methods
- If these do not return approximately the same result, there is insufficient information available
- Some action should be taken to find out more in order to make more accurate estimates
- Pricing to win is sometimes the only applicable method

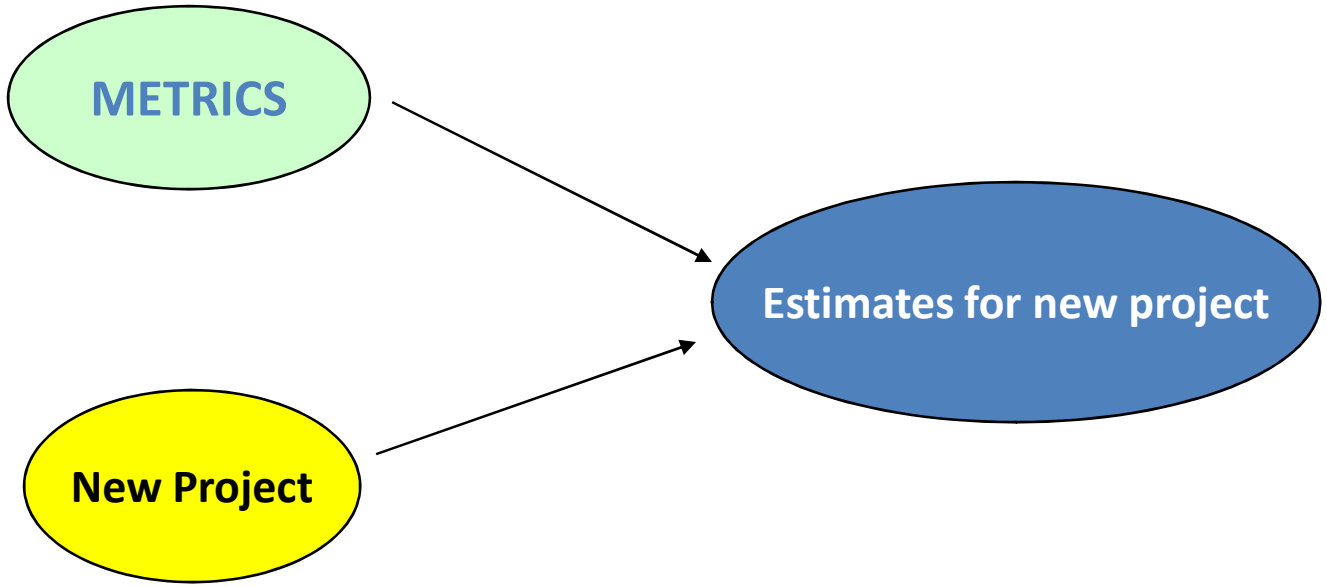
Algorithmic Cost Modelling

- Most of the work in the cost estimation field has focused on algorithmic cost modelling.
- Costs are analysed using mathematical formulas linking costs or inputs with METRICS to produce an estimated output.
- The formula is based on the analysis of historical data.
- The accuracy of the model can be improved by calibrating the model to your specific development environment, (which basically involves adjusting the weighting parameters of the metrics).

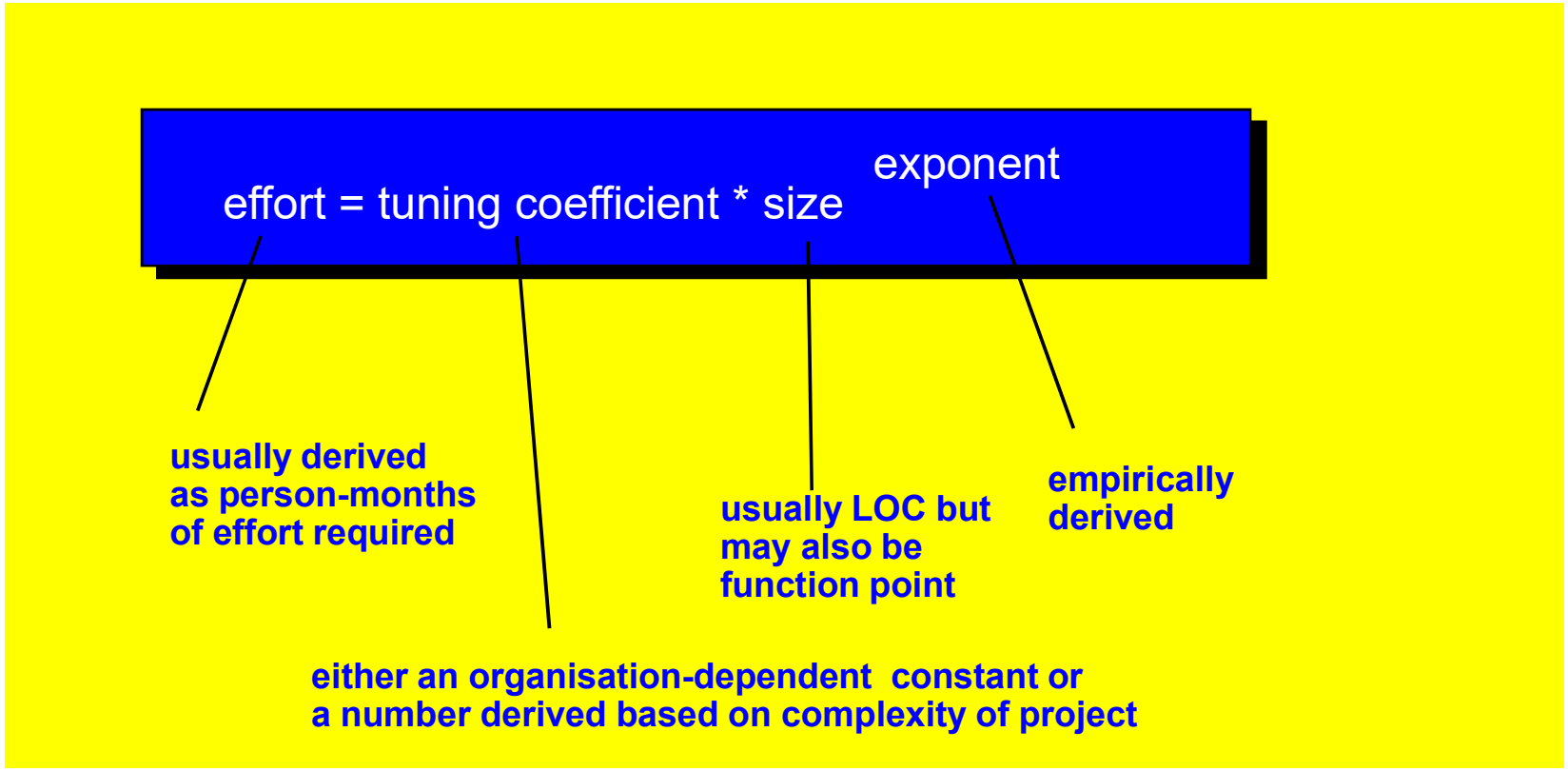
Building Metrics from measurements



New Project estimation using available Metrics



Empirical Estimation Models - Algorithmic Cost Modelling



Algorithmic Cost Modelling

- $\text{Effort} = A \times \text{Size}^B \times M$
  - A is an organisation-dependent constant
  - B reflects the nonlinearity (disproportionate) effort for large projects
  - M is a multiplier reflecting product, process and people attributes

- Most commonly used product attribute for cost estimation is code size (LOC)
- Most models are basically similar but with different values for A, B and M

Estimation Accuracy

- The size of a software system can only be known accurately when it is finished
- Several factors influence the final size
  - Use of COTS and components
  - Programming language
  - Distribution of system
- As the development process progresses then the size estimate becomes more accurate

Estimate Uncertainty