The COCOMO Cost model
Constructive Cost Model

- An empirical model based on project experience
- COCOMO'81 is derived from the analysis of 63 software projects in 1981.
- Well-documented, ‘independent’ model which is not tied to a specific software vendor

- COCOMO II (2000) takes into account different approaches to software development, reuse, etc.

COCOMO II

- COCOMO II is a 3-level model that allows increasingly detailed estimates to be prepared as development progresses
  - Early prototyping level
    - Estimates based on object-points and a simple formula is used for effort estimation
  - Early design level
    - Estimates based on function-points that are then translated to LOC
    - Includes 7 cost drivers
  - Post-architecture level
    - Estimates based on lines of source code or function point
    - Includes 17 cost drivers

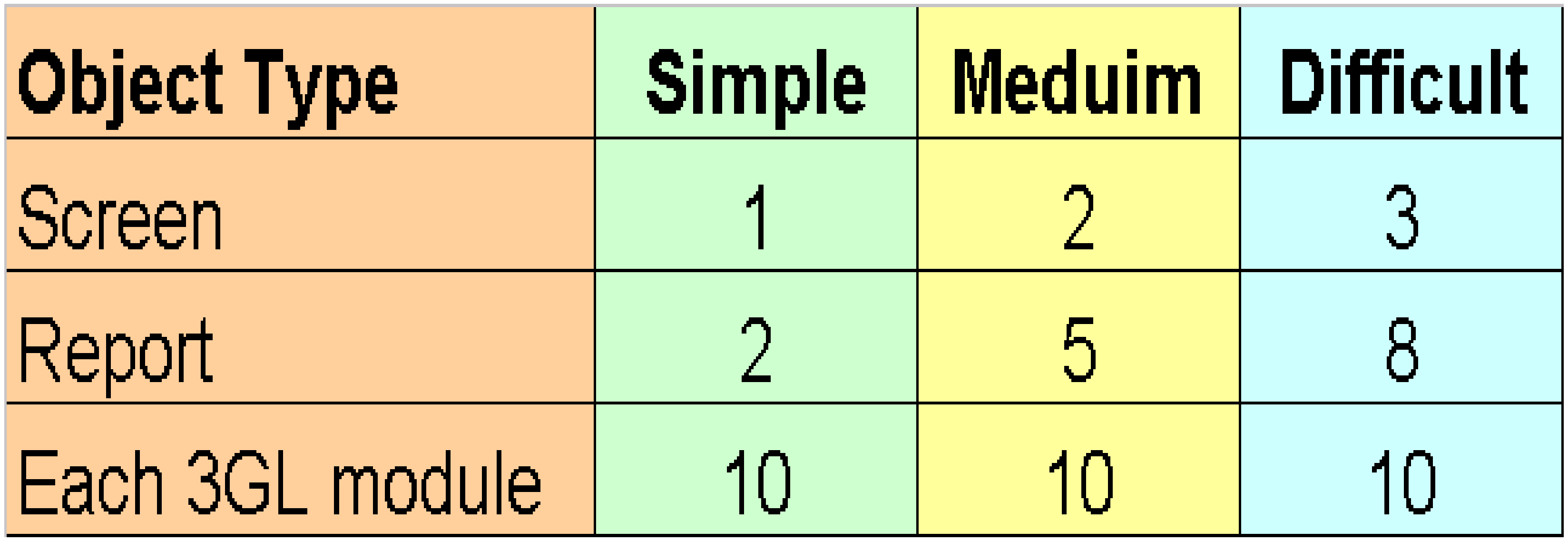
COCOMO II Early prototyping level Object-Points

- Suitable for projects built using modern GUI-builder tools
  - Based on Object-Points
- Supports prototyping projects and projects where there is extensive reuse
- Based on standard estimates of developer productivity in object points/month
- Takes CASE tool use into account
- Formula is
  - $PM = (NOP \cdot (1 - \%reuse / 100)) / PROD$
  - PM is the effort in person-months, NOP is the number of object points and PROD is the productivity

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 considered as a weighted estimate of 3 elements:
  - 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

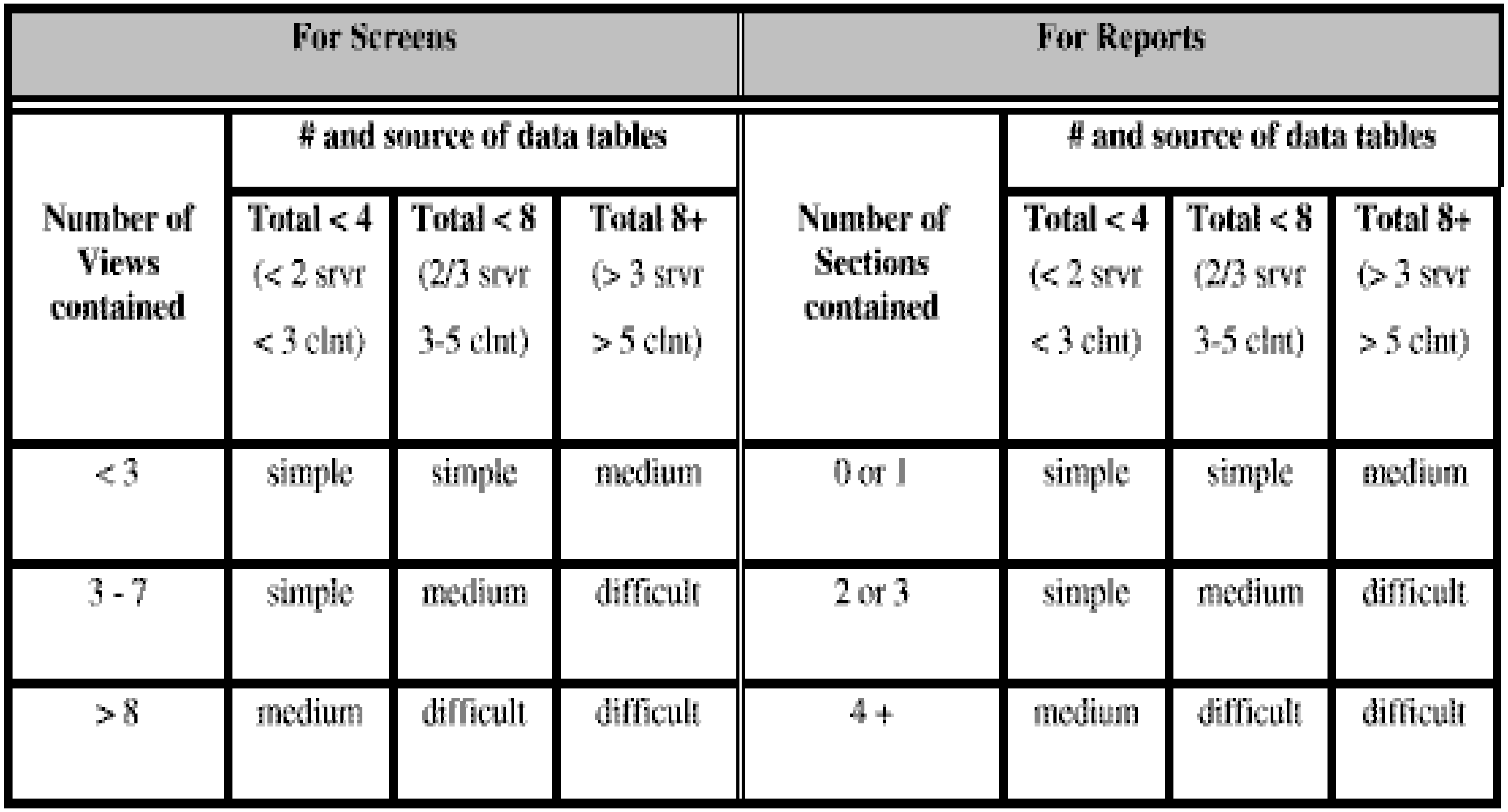
Object-Points – Weighting



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

Object-Points: Complexity Levels

- *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 easy to estimate
  - simply concerned with screens, reports and 3GL modules
- At an early point in the development process:
  - Object-points can be early 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: PROD
  - measured 4 - 50 object points/person-month
  - depends on tool support and developer capability

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 Object-points (NOP=840) & Productivity is very high (= 50 object points/person-month)
  - Then, Effort = 840/50 = (16.8) = 17 p-m

Adjustment for % of Reuse

- % reuse: the % of screens, reports, & 3GL modules reused from previous applications, pro-rated by degree of reuse
  - Adjusted NOP = NOP * (1 - % reuse / 100)
  - Adjusted NOP: New NOP
  - Example:
    - An application contains 840 OP, of which 20% can be supplied by existing components.

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

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

Object-Point Estimation Procedure

1. Assess object-counts in the system: number of screens, reports, & 3GL.
2. Assess complexity level for each object (use table): simple, medium and difficult.
3. Calculate “NOP” the object-point count of the system: add all weights for all object instances
4. Estimate the % of reuse and compute the adjusted NOP “New Object Points “ to be developed
5. Determine the productivity rate PROD (use metrics table)
6. Compute the adjusted effort PM = adjusted NOP / PROD

Object-Point Estimation Example

Assessment of a software system shows that:

- The system includes
  - 6 screens: 2 simple + 3 medium + 1 difficult
  - 3 reports: 2 medium + 1 difficult
  - 2 3GL components
- 30 % of the objects could be supplied from previously developed components
- Productivity is high

Compute the estimated effort PM ‘Person-months’ needed to develop the system.

OP Estimation Example: Solution

- Object counts:

2 simple screens	x 1	=	2
3 medium screens	x 2	=	6
1 difficult screen	x 3	=	3
2 medium reports	x 5	=	10
1 difficult report	x 8	=	8
2 3GL components	x 10	=	20
NOP			49

OP Estimation Example: Solution

- Adjusted NOP 'New NOP' = NOP * (1 - % reuse / 100)
  = 49 * (1- (30/100))
  = (34.3)
  = 35

- For high productivity (metric table): PROD = 25 OP/P-M

- Estimated effort Person-Month = Adjusted NOP / PROD
  = 35/ 25
  = 1.4 P-M