Empirical Estimation Models

Empirical Estimation Models

Introduction

- An estimation model for computer software uses derived formulas to predict effort as a function of LOC or FP.
- The empirical data that support most estimation models are derived from a limited sample of projects.
- No estimation model is appropriate for all classes of software and in all development environments. Therefore, you should use the results obtained from such models judiciously.
- An estimation model should be calibrated to reflect local conditions

The Structure of Estimation Models

- A typical estimation model is derived using regression analysis on data collected from past software projects.

$$E = A + B \times (e_v)^C$$

- where A, B, and C are empirically derived constants, E is an effort in person-months, and e_v is the estimation variable (either LOC or FP).

Empirical Estimation Models

The Structure of Estimation Models Cont'd

- The following are the LOC-oriented estimation models proposed in the literature.

Model Name	Equation
Walston-Felix model	$E = 5.2 * (KLOC)^{0.91}$
Bailey-Basili model	$E = 5.5 + 0.73 * (KLOC)^{1.16}$
Boehm simple model	$E = 3.2 * (KLOC)^{1.05}$
Doty model for KLOC >9	$E = 5.288 * (KLOC)^{1.047}$

- The following are the FP-oriented estimation models proposed in the literature.

Model Name	Equation
Albrecht and Gaffney model	$E = -91.4 + 0.355FP$
Kemerer model	$E = -37 + 0.96FP$
Small project regression model	$E = -12.88 + 0.405FP$

Empirical Estimation Models

COCOMO Model

- COCOMO (Constructive Cost Model) is a regression model based on the number of Lines of Code (LOC).
- It is a procedural cost estimation model for software projects.
- This model is used for predicting the various parameters associated with developing a project such as size, effort, cost, and time.
- COCOMO was proposed by Barry Boehm in 1970 and is based on the study of 63 projects.

Effort and schedule parameters are outcomes of the COCOMO model.

- Effort: Amount of labor (staff) that will be required to complete a task. It is measured in person-months units.
- Schedule: means the amount of time required for the completion of the job, which is proportional to the effort. It is measured in the units of time such as weeks, and months.

Empirical Estimation Models

Boehm's categorizes the projects

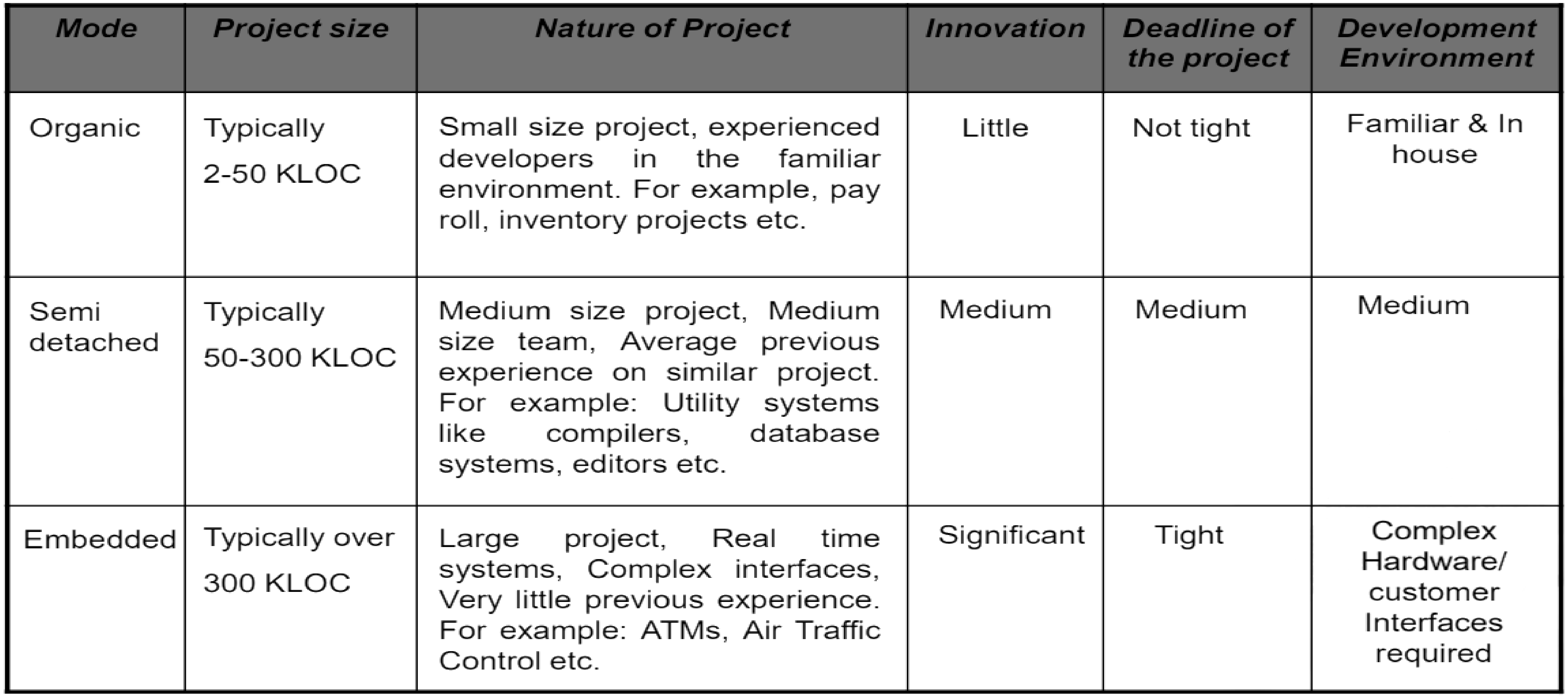
1. Organic: If the team size required to develop a software project is small then the software project is called as Organic project. In this the problem is well understood and has been solved in the past and also the team members have a nominal experience regarding the problem.

2. Semi-detached: If the main features such as team size, experience, and knowledge of the various programming environment lie in between Organic and Embedded then the software project is called a Semi-detached type project. Eg: Compilers or different Embedded Systems.

3. Embedded: Embedded software projects require the highest level of complexity, creativity, and experience. This type of software requires a larger team size than the other two models and also the developers need to be sufficiently experienced and creative to develop such complex models.

Empirical Estimation Models

Boehm's categorizes the projects



Mode	*Project size*	*Nature of Project*	*Innovation*	*Deadline of the project*	*Development Environment*
Organic	Typically 2-50 KLOC	Small size project, experienced developers in the familiar environment. For example, pay roll, inventory projects etc.	Little	Not tight	Familiar & In house
Semi detached	Typically 50-300 KLOC	Medium size project, Medium size team, Average previous experience on similar project. For example: Utility systems like compilers, database systems, editors etc.	Medium	Medium	Medium
Embedded	Typically over 300 KLOC	Large project, Real time systems, Complex interfaces, Very little previous experience. For example: ATMs, Air Traffic Control etc.	Significant	Tight	Complex Hardware/ customer Interfaces required

Empirical Estimation Models

Types of COCOMO Models

There are three types of COCOMO models:

1. Basic COCOMO Model
2. Intermediate COCOMO Model
3. Detailed COCOMO Model

COCOMO model levels:

- Basic-predicted software size (lines of code) was used to estimate development effort.
- Intermediate - predicted software size (lines of code), plus a set of 15 subjectively assessed 'cost drivers' was used to estimate development effort.
- Advanced - on top of the intermediate model, the advanced model allows phase-based cost driver adjustments and some adjustments at the module, component, and system levels.

Empirical Estimation Models

1. Basic COCOMO model

- The Basic COCOMO can be used for quick and slightly rough calculations of Software Costs.

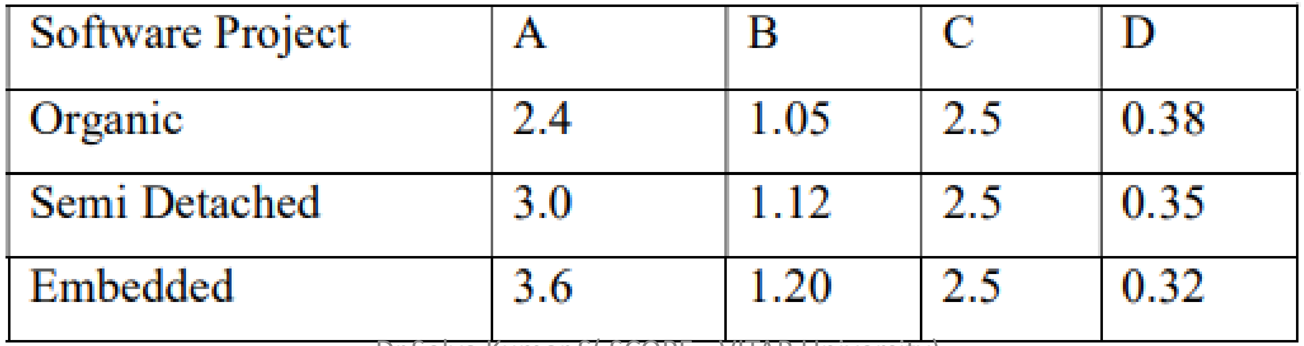
- The following formulas are used to calculate effort and time estimates.

  Effort (E) = $A*(KLOC)^B$

  Time (T) = $C*(Effort)^D$

  Persons required = Effort / Time

- The constant values A, B, C and D for the Basic Model for the different categories of software projects.



Software Project	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

Empirical Estimation Models

1. Example of Basic COCOMO model

- Supposed that a project was estimated to be 400 KLOC. Calculate effort and development time for each of three modes i.e. Organic, semi-detached and embedded.

- Effort (E) = $A*(KLOC)^B$
- Time (T) = $C*(Effort)^D$
- Estimated size of project = 400 KLOC.

Organic mode

$E = 2.4 * (400)^{1.05}$ = 1295.31 Person-Months

$T = 2.5 * (1295.31)^{.38}$ = 38.07 Months

Semi-detached mode

$E = 3.0 * (400)^{1.12}$ = 1295.31 Person-Months

$T = 2.5 * (12462.79)^{.35}$ = 38.45 Months

Embedded Mode

$E = 3.6 * (400)^{1.2}$ = 4772.81 Person-Months

$T = 2.5 * (4772.81)^{.32}$ = 38 Months

Empirical Estimation Models

Try this

- A project size of 200 KLOC is to be developed. The software development team has average experience on similar types of projects. The project schedule is not very tight. Calculate the effort, development time, average staff size, and productivity of the project.

Empirical Estimation Models

2. Intermediate COCOMO Model

- In the intermediate COCOMO model, various other factors such as reliability, experience, and Capability are considered along with LOC to estimate effort and time schedule.
- These factors are known as Cost Drivers and the Intermediate Model utilizes 15 such drivers for cost estimation.

(i) Product attributes:

1. Required software reliability extent
2. Size of the application database
3. The complexity of the product

(ii) Hardware attributes:

4. Run-time performance constraints
5. Memory constraints
6. The volatility of the virtual machine environment
7. Required turnabout time

(iii) Personnel attributes:

8. Analyst capability
9. Software engineering capability
10. Applications experience
11. Virtual machine experience
12. Programming language experience

(iv) Project attributes:

13. Use of software tools
14. Application of software engineering methods
15. Required development schedule

Empirical Estimation Models

2. Intermediate COCOMO Model Cont'd

Cost drivers Number	Ratings					
	Very low	Low	Nominal	High	Very high	Extra high
1	0.75	0.88	1.00	1.15	1.40	-
2	-	0.94	1.00	1.08	1.16	-
3	0.70	0.85	1.00	1.15	1.30	1.65
4	-	-	1.00	1.11	1.30	1.66
5	-	-	1.00	1.06	1.21	1.56
6	-	0.87	1.00	1.15	1.30	-
7	-	0.87	1.00	1.07	1.15	-
8	1.46	1.19	1.00	0.86	0.71	-
9	1.29	1.13	1.00	0.91	0.82	-
10	1.42	1.17	1.00	0.86	0.70	-
11	1.21	1.10	1.00	0.90	-	-
12	1.14	1.07	1.00	0.95	-	-
13	1.24	1.10	1.00	0.91	0.82	-
14	1.24	1.10	1.00	0.91	0.83	-
15	1.23	1.08	1.00	1.04	1.10	-

The Intermediate COCOMO model formulas are

Effort (E) = $(A*(KLOC)^{B}) * EAF$.

Time (T) = $C*(Effort)^{D}$

Persons required = Effort / Time

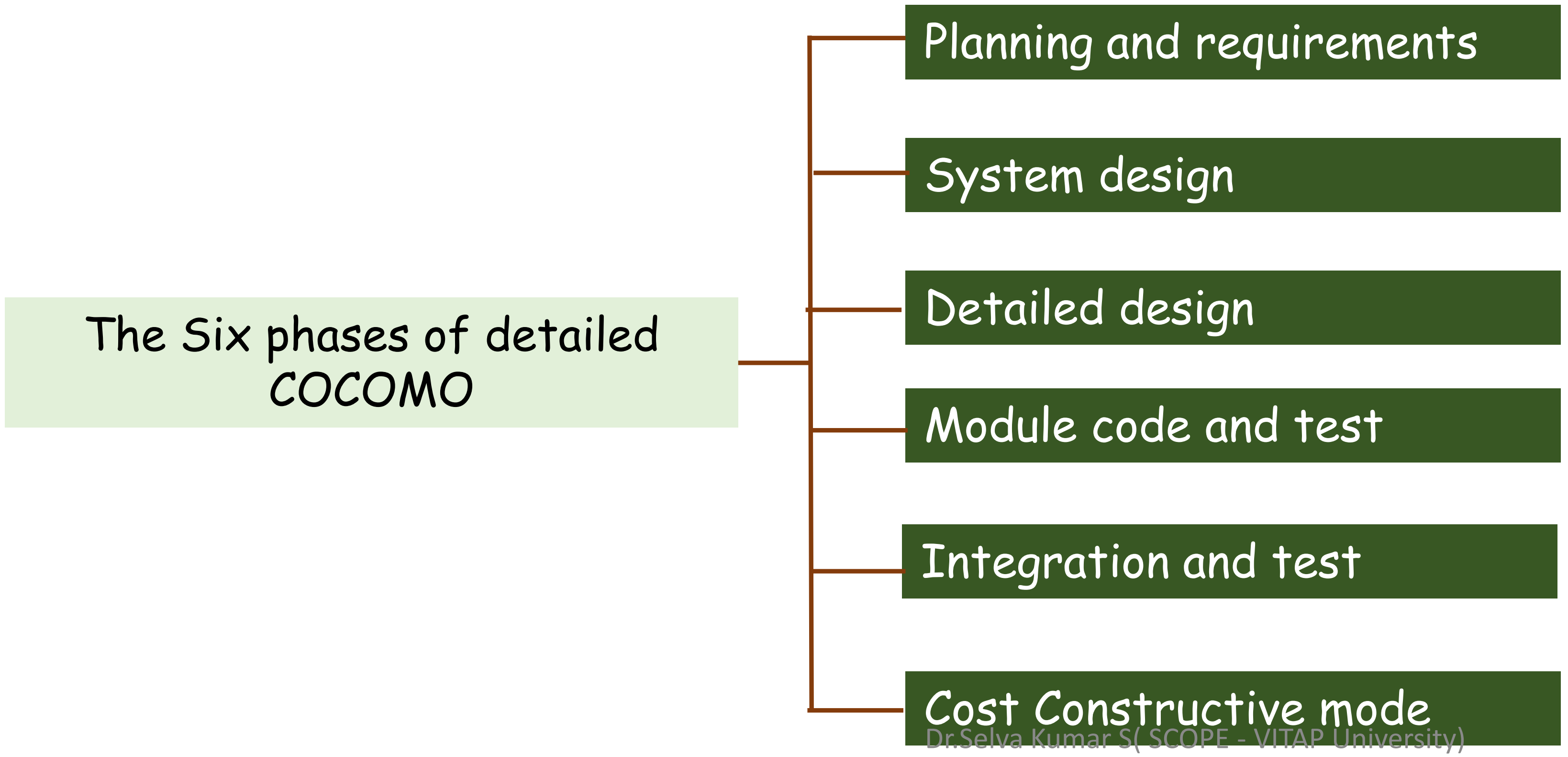
- The constant values A, B, C and D for the Intermediate Model for the different categories of software projects.

Software Project	A	B	C	D
Organic	3.2	1.05	2.5	0.38
Semi Detached	3.0	1.12	2.5	0.35
Embedded	2.8	1.20	2.5	0.32

Empirical Estimation Models

3. Detailed COCOMO Model

- The detailed COCOMO incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step of the software engineering process. The detailed model uses different effort multipliers for each cost driver attribute.



- The effort is calculated as a function of program size and a set of cost drivers are given according to each phase of the software lifecycle.

Empirical Estimation Models

COCOMO II Model

COCOMO II is a hierarchy of estimation models that address the following areas

- Application composition model: Used during the early stages of software engineering, when prototyping of user interfaces, consideration of software and system interaction, assessment of performance, and evaluation of technology maturity are paramount.
- Early design stage model: Used once requirements have been stabilized and basic software architecture has been established.
- Post-architecture-stage model: Used during the construction of the software

- COCOMO II models require sizing information. Three different sizing options are available as part of the model hierarchy. They are object points, function points, and lines of source code.

Empirical Estimation Models

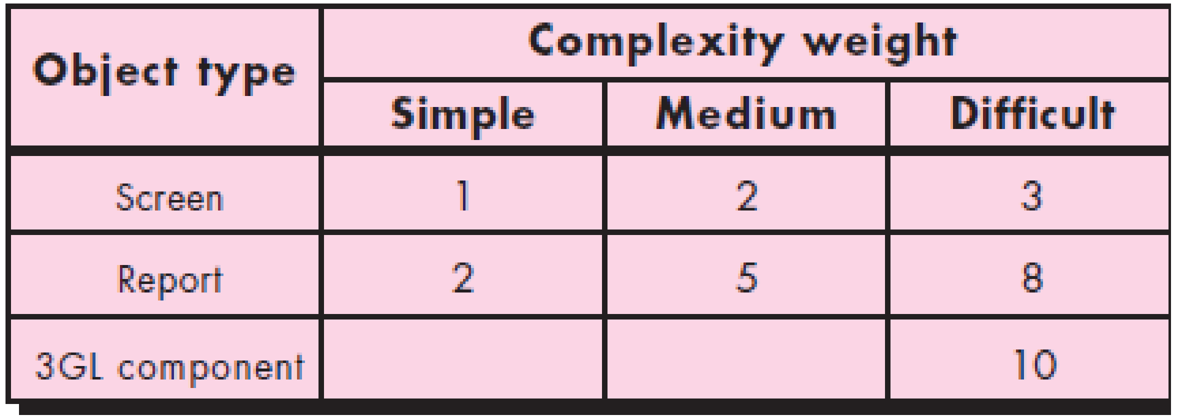
COCOMO II Model Cont'd

- The COCOMO II application composition model uses object points.
- Like function points, the object point is an indirect software measure that is computed using counts of the number of

  (1) screens (at the user interface)

  (2) reports, and

  (3) components likely to be required to build the application

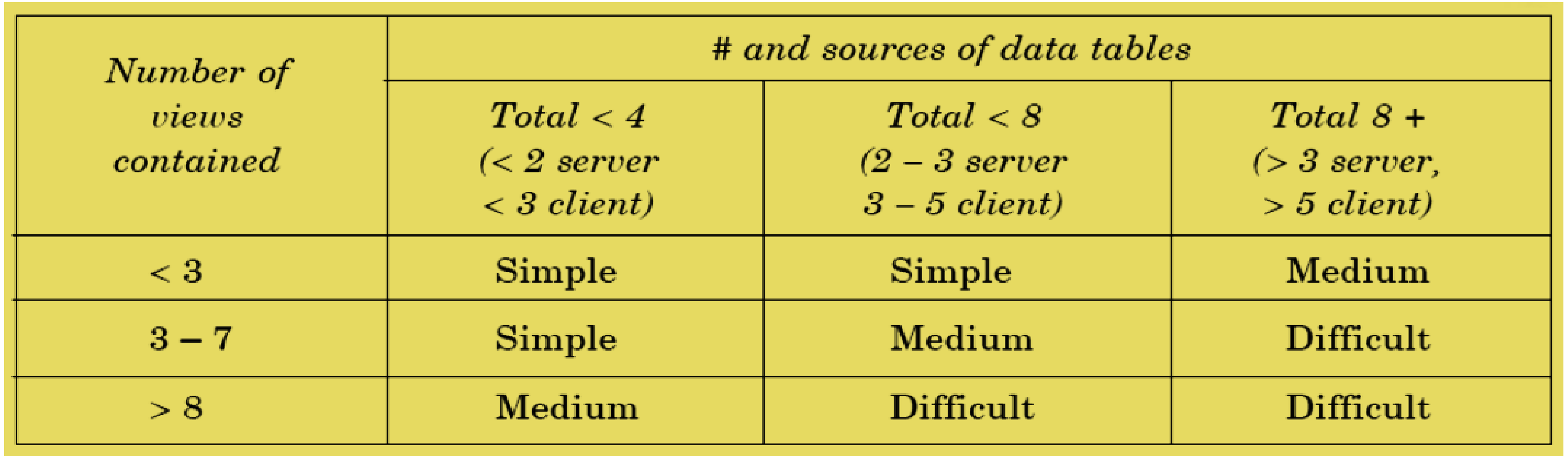


Object type	Complexity weight		
	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL component			10

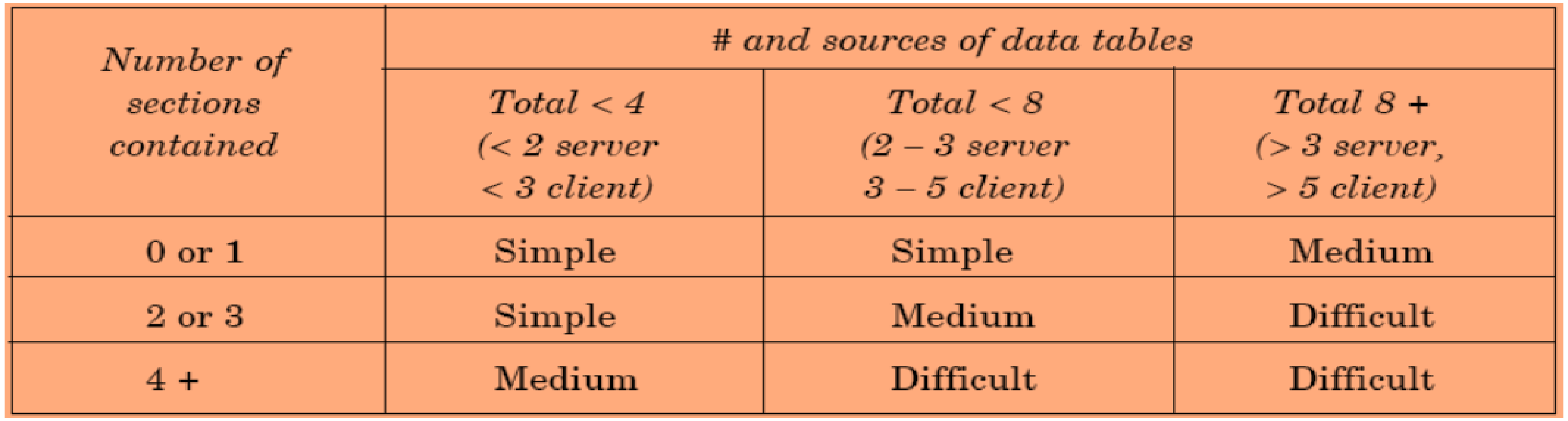
- The object point count is determined by multiplying the original number of object instances by the weighting factor.

Empirical Estimation Models

COCOMO II Model Cont'd



Number of views contained	*# and sources of data tables*		
	Total < 4 (< 2 server < 3 client)	*Total < 8 (2 – 3 server 3 – 5 client)*	*Total 8 + (> 3 server, > 5 client)*
< 3	Simple	Simple	Medium
3 – 7	Simple	Medium	Difficult
> 8	Medium	Difficult	Difficult



Number of sections contained	*# and sources of data tables*		
	Total < 4 (< 2 server < 3 client)	*Total < 8 (2 – 3 server 3 – 5 client)*	*Total 8 + (> 3 server, > 5 client)*
0 or 1	Simple	Simple	Medium
2 or 3	Simple	Medium	Difficult
4 +	Medium	Difficult	Difficult

Empirical Estimation Models

COCOMO II Model Cont'd

- When component-based development or general software reuse is to be applied, the percent of reuse (%reuse) is estimated and the object point count is adjusted:

NOP = (Object points) * [(100 - % reuse)/100]

- Where NOP = New Object Points. Now Productivity rate is defined based on the computed NOP value to derive an estimate of effort.

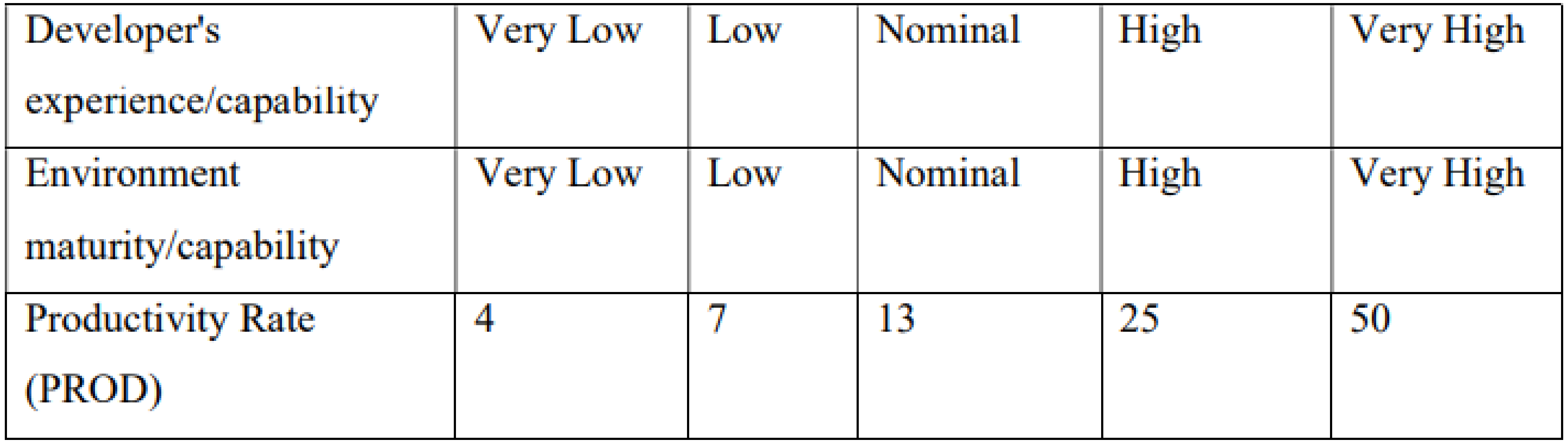
Productivity rate (PROD) = NOP/ Person-Month

Estimated effort = NOP/PROD

Empirical Estimation Models

COCOMO II Model Cont'd

The productivity rate for different levels of developer experience and development environment maturity.



Developer's experience/capability	Very Low	Low	Nominal	High	Very High
Environment maturity/capability	Very Low	Low	Nominal	High	Very High
Productivity Rate (PROD)	4	7	13	25	50

Empirical Estimation Models

COCOMO II Model Cont’d

Example

i) The application has 4 screens with 4 views each and 7 data tables for 3 servers and 4 clients.

ii) The application may generate two reports of 6 sections each from 7 data tables for two servers and 3 clients. There is a 10% reuse of object points.

The developer’s experience and capability in a similar environment is low. The maturity of the organization in terms of capability is also low.

Calculate the object point count, new object points, and effort to develop such a project

Number of screens = 4 with 4views each

Number of reports = 2 with 6 sections each.

The Complexity weights, Object point count = 4 * 2 + 2 * 8 = 24

(screen medium complexity = 2, report difficult complexity =8)

Empirical Estimation Models

COCOMO II Model Cont'd

Example Cont'd

New Object Points (NOP) = (object points) * [(100 - %reuse)/100]

= 24 * (100-10)/100

= 24 * 90/100 = 21.6

The low value of productivity (PROD) = 7

Efforts in Person – Months = **NOP/PROD** = 21.6 / 7 = 3.086 = 3 Person-months.

Empirical Estimation Models

The Software Equation (Putnam-1992)

- The software equation is a dynamic multivariable model that assumes a specific distribution of effort over the life of a software development project.
- This model has been derived from productivity data collected for over 4000 contemporary software projects.

$$E = \frac{\text{LOC} \times B^{0.333}}{P^3} \times \frac{1}{t^4}$$

where

E - effort in person-months or person-years

t - project duration in months or years

B - "special skills factor"

P - "productivity parameter".

- Typical values might be P 2000 for the development of real-time embedded software, P 10,000 for telecommunication and systems software, and P 28,000 for business systems applications

Empirical Estimation Models

The Software Equation (Putnam-1992)

- Minimum development time is defined as

$$t_{\min} = 8.14 \frac{\text{LOC}}{P^{0.43}} \text{ in months for } t_{\min} > 6 \text{ months}$$

t – Represent in years.

$$E = 180\ Bt^3 \text{ in person-months for } E \geq 20 \text{ person-months}$$

For Example, *P* 12,000 (Recommended value for scientific software), and suppose the total LOC of the project is 33,200 then

$$t_{\min} = 8.14 \times \frac{33{,}200}{12{,}000^{0.43}} = 12.6 \text{ calendar months}$$

$$E = 180 \times 0.28 \times (1.05)^3 = 58 \text{ person-months}$$

Estimation for Object-oriented Projects

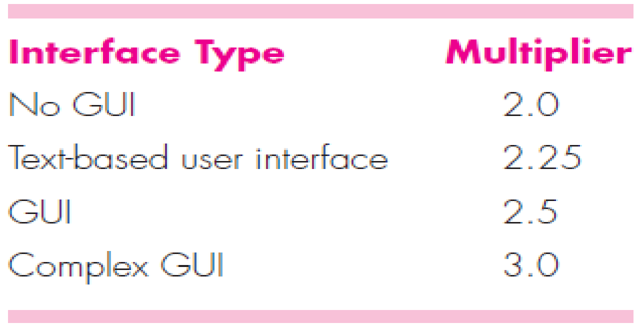
Lorenz and Kidd suggested the following approach:

1. Develop estimates using effort decomposition, FP analysis, and any other method that is applicable to conventional applications.

2. Using the requirements model, develop use cases and determine a count. Recognize that the number of use cases may change as the project progresses.

3. From the requirements model, determine the number of key classes.

4. Categorize the type of interface for the application and develop a multiplier for support classes.



Interface Type	Multiplier
No GUI	2.0
Text-based user interface	2.25
GUI	2.5
Complex GUI	3.0

Multiply the number of key classes (step 3) by the multiplier to obtain an estimate for the number of support classes

5. Multiply the total number of classes (key + support) by the average number of work units per class. Lorenz and Kidd suggest 15 to 20 person-days per class.

6. Cross-check the class-based estimate by multiplying the average number of work units per use case.

Specialized Estimation Techniques

1. Estimation for Agile Development

- Estimation for agile projects uses a decomposition approach

1. Each user scenario (the equivalent of a mini-use case created at the very start of a project by end users or other stakeholders) is considered separately for estimation purposes.

2. The scenario is decomposed into the set of software engineering tasks that will be required to develop it.

3a. The effort required for each task is estimated separately. Note: Estimation can be based on historical data, an empirical model, or "experience."

3b. Alternatively, the "volume" of the scenario can be estimated in LOC, FP, or some other volume-oriented measure (e.g., use-case count).

4a. Estimates for each task are summed to create an estimate for the scenario.

4b. Alternatively, the volume estimate for the scenario is translated into effort using historical data.

5. The effort estimates for all scenarios that are to be implemented for a given software increment are summed to develop the effort estimate for the increment.

Specialized Estimation Techniques Cont'd

2. Estimation for WebApp Projects

- *Inputs* are each input screen or form (for example, CGI or Java), each maintenance screen, and if you use a tab notebook metaphor anywhere, each tab.
- *Outputs* are each static Web page, each dynamic Web page script (for example, ASP, ISAPI, or other DHTML script), and each report (whether Web based or administrative in nature).
- *Tables* are each logical table in the database plus, if you are using XML to store data in a file, each XML object (or collection of XML attributes).
- *Interfaces* retain their definition as logical files (for example, unique record formats) into our out-of-the-system boundaries.
- *Queries* are each externally published or use a message-oriented interface. A typical example is DCOM or COM external references.

The Make/Buy Decision

- It may be more cost-effective to acquire a piece of software rather than develop it.
- Decision tree analysis provides a systematic way to sort through the make/buy decision.
- As a rule, outsourcing software development requires more skillful management than in-house development of the same product.

Decision Tree helps "make / Buy Decisions"

Expected cost = Σ (path probability)$_i$ $\times$ (estimated path cost)$_i$

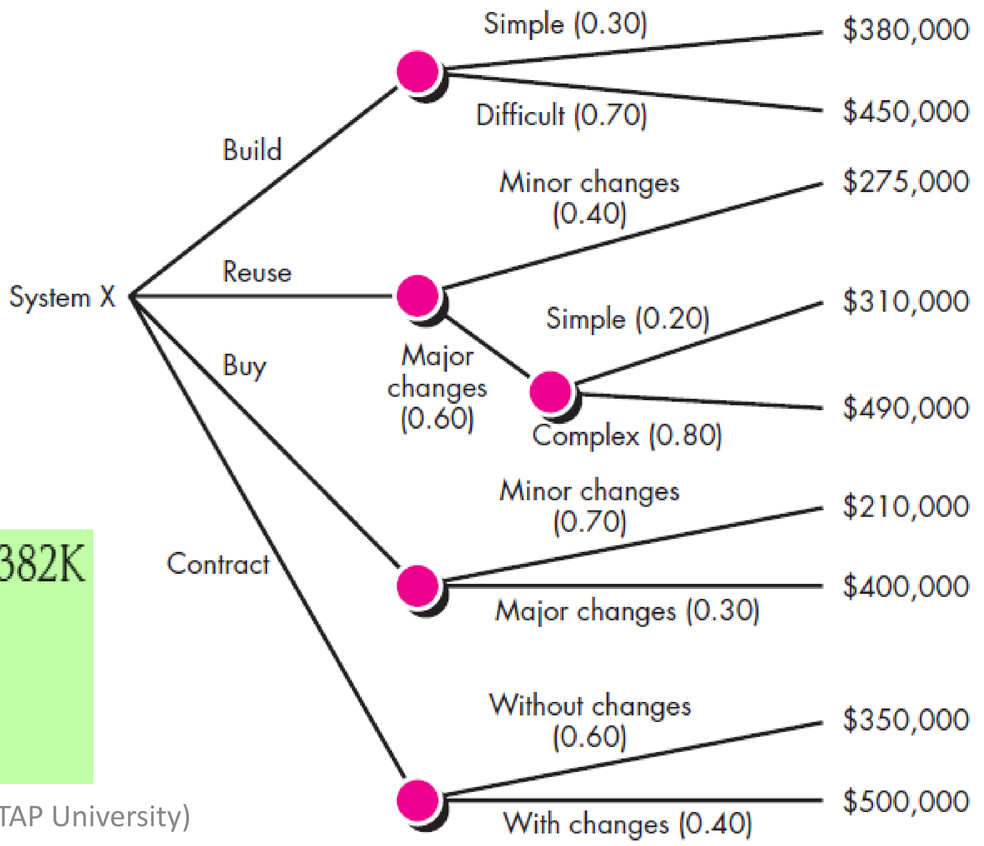
where i is the decision tree path. For the build path,

Expected cost$_{\text{build}}$ = 0.30 (\$380K) + 0.70 (\$450K) = \$429K

Expected cost$_{\text{reuse}}$ = 0.40 (\$275K) + 0.60 [0.20 (\$310K) + 0.80 (\$490K)] = \$382K

Expected cost$_{\text{buy}}$ = 0.70 (\$210K) + 0.30 (\$400K) = \$267K

Expected cost$_{\text{contract}}$ = 0.60 (\$350K) + 0.40 (\$500K) = \$410K


System X
Build
Simple (0.30)
$380,000
Difficult (0.70)
$450,000
Reuse
Minor changes (0.40)
$275,000
Major changes (0.60)
Simple (0.20)
$310,000
Complex (0.80)
$490,000
Buy
Minor changes (0.70)
$210,000
Major changes (0.30)
$400,000
Contract
Without changes (0.60)
$350,000
With changes (0.40)
$500,000