



University of Southern California
Center for Software Engineering

COCOMO II Model

Brad Clark

CSE Research Associate

15th COCOMO/SCM Forum

October 22, 1998

Brad@Software-Metrics.com



COCOMO II Model Overview

- ➔ • COCOMO II Overview**
- Sizing the Application**
- Estimating Effort**
- Estimating Schedule**
- Understanding model workings**
- Estimating Software Maintenance**



Early Design and Post-Arch Models

- **Nominal-Schedule Estimated Effort (PM_{NS}):**
(excludes Required Development Schedule cost driver)

$$PM_{NS} = A \times (\text{Size})^E \times \prod_{i=1}^n EM_i$$

$$\text{where } E = B + 0.01 \times \sum_{j=1}^5 SF_j$$

- **A = 2.94** **B = 0.91**
- **Size** (thousands of lines of code, KSLOC, or function points)
- **EM: Effort Multipliers** (6 for ED, 16 for PA)
- **SF: Scale Factors** (5 for both models)



Early Design and Post-Arch Models

- **Nominal-Schedule Estimated Duration ($TDEV_{NS}$):**
(excludes Required Development Schedule cost driver)

$$TDEV_{NS} = C \times (PM_{NS})^F$$

$$\text{where } F = D + 0.2 \times 0.01 \times \sum_{j=1}^5 SF_j$$

- **$C = 3.67$**
- **PM_{NS} = from nominal-schedule effort estimation**
- **$D = 0.28$**
- **SF: Scale Factors (5 for both models)**



Post-Arch Model Example

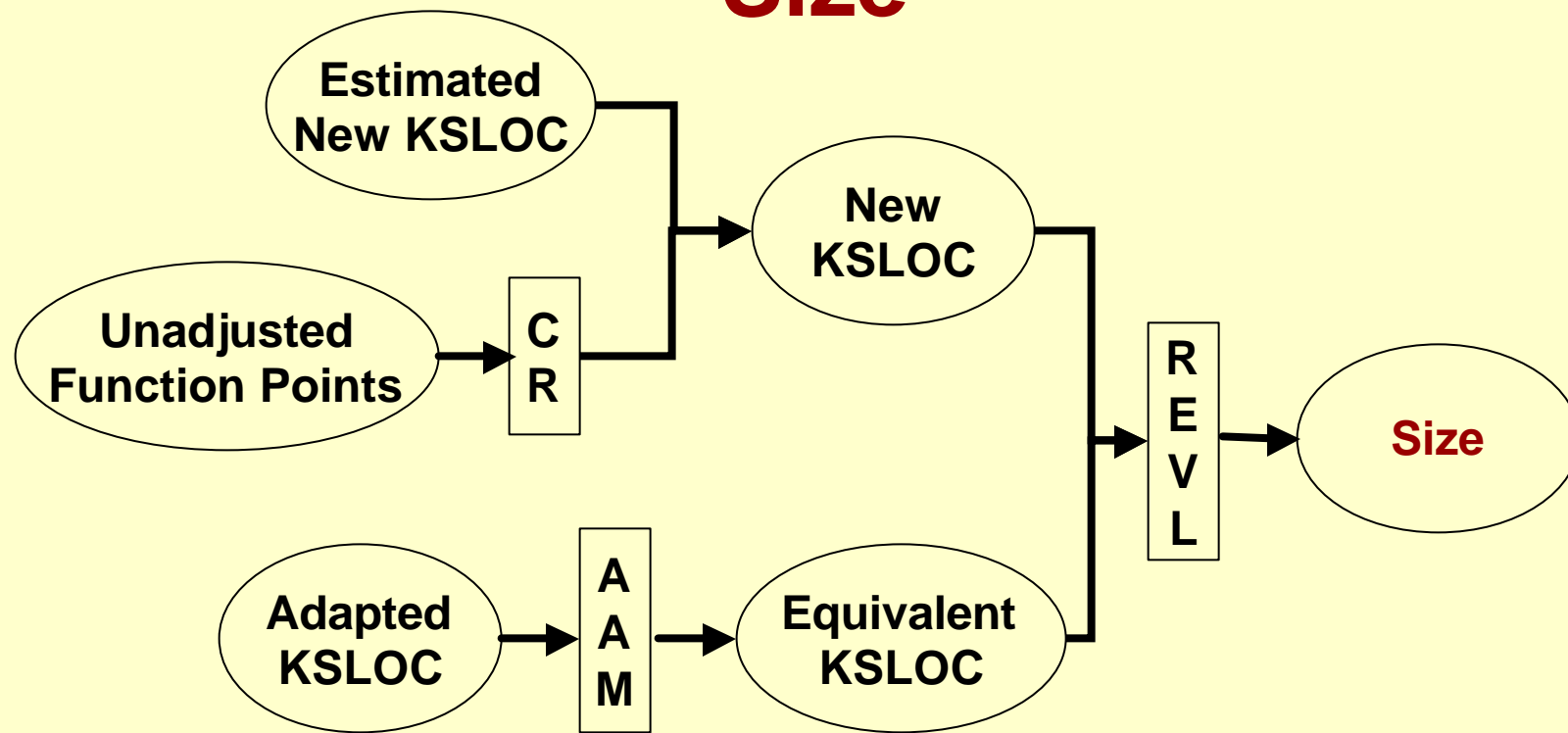
- **Suppose we had a 100 KSLOC, average project**
 - **Sum EM's = 1.0 (all nominal ratings)**
 - **Sum SF's = 24 (mostly low ratings)**
- **$PM_{NS} = 2.94(100)^{(0.91+0.01*24)} = 586.61$ person months**
- **$TDEV_{NS} = 3.67(586.6)^{(0.28+0.2*0.01*24)} = 29.7$ months**
- **Average number of staff = $PM_{NS}/TDEV_{NS} = 19.75$ people**



COCOMO II Model Overview

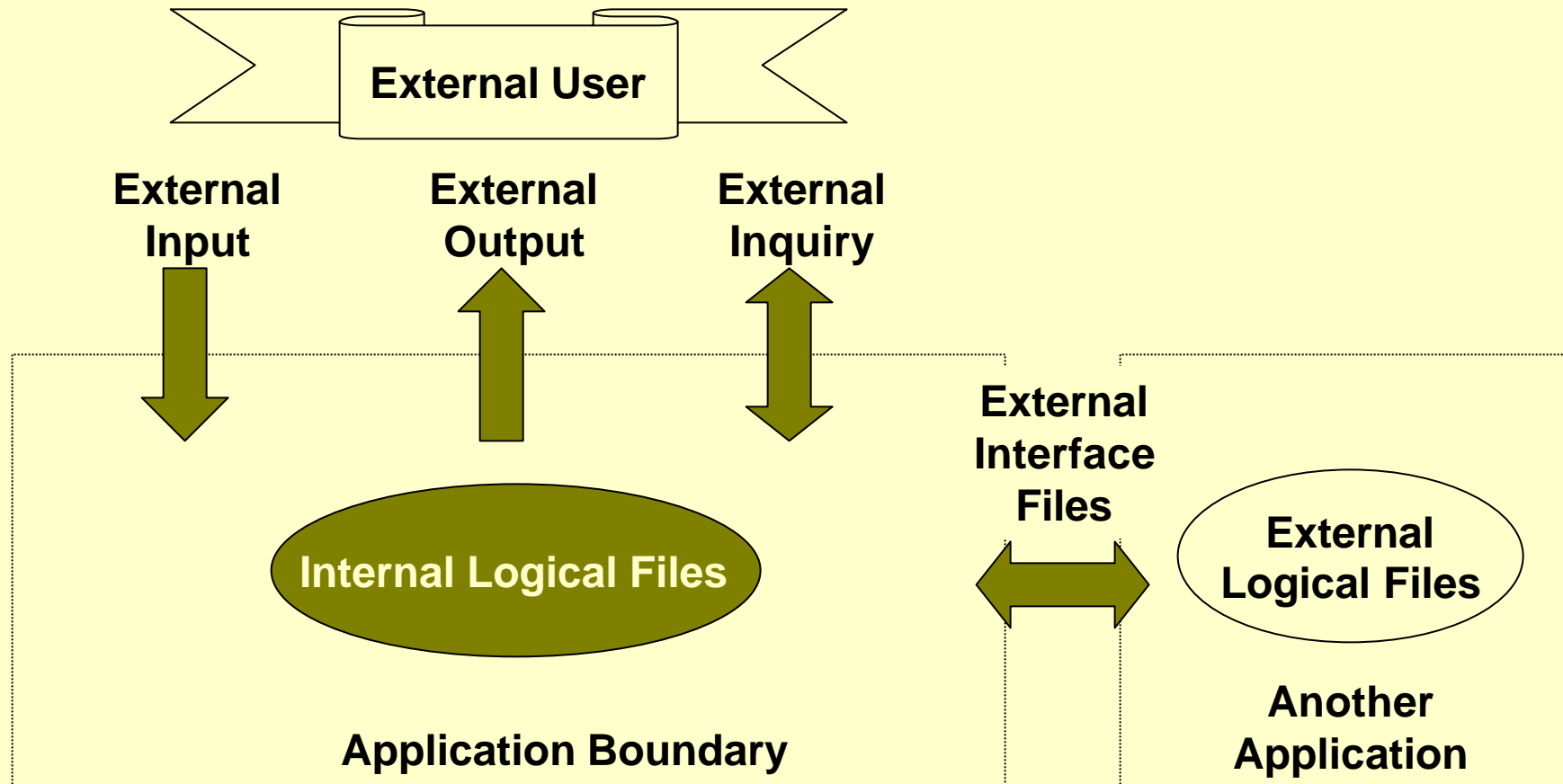
- **COCOMO II Overview**
- ➔ • **Sizing the Application**
- **Estimating Effort**
- **Estimating Schedule**
- **Understanding model workings**
- **Estimating Software Maintenance**

Size



KSLOC: Thousands of Source Lines of Code
CR: Conversion Ratios
REVL: Requirements Evolution
AAM: Adaptation Adjustment Modifiers

Function Point Definition





Some Function Point Conversion Ratios

<u>Language</u>	<u>SLOC / UFP</u>
Ada 83	71
Ada 95	49
AI Shell	49
APL	32
Assembly - Basic	320
Assembly - Macro	213
Basic – ANSI	64
Basic – Compiled	91
Basic – Visual	32
C	128
C++	55
Cobol (ANSI 85)	91
...	...



SLOC Definition Considerations

- Whether to **include** or **exclude**
 - executable and/or non-executable code statements
 - code produced by programming, copying without change, automatic generation, and/or translation
 - newly developed code and/or previously existing code
 - product-only statements or also include support code
 - counts of delivered and/or non-delivered code
 - counts of operative code or include dead code
 - replicated code
- When does the code get counted
 - at estimation, at design, at coding, at unit testing, at integration, at test readiness review, at system test complete



Using Preexisting Code

- **Reused Code**
 - Preexisting code that is treated as a black-box and plugged into the product
- **Adapted Code**
 - Preexisting code that is treated as white-box and is modified for use with the product
- **The size of reused and adapted code is adjusted to be its **equivalent** in new code using Adaptation Adjustment Modifiers (AAM)**
 - Based on additional effort it takes to modify the code for inclusion in the product
 - Adaptation with function points and source lines of code is the same for Early Design and Post-Arch Models



Non-Linear Reuse Size Model

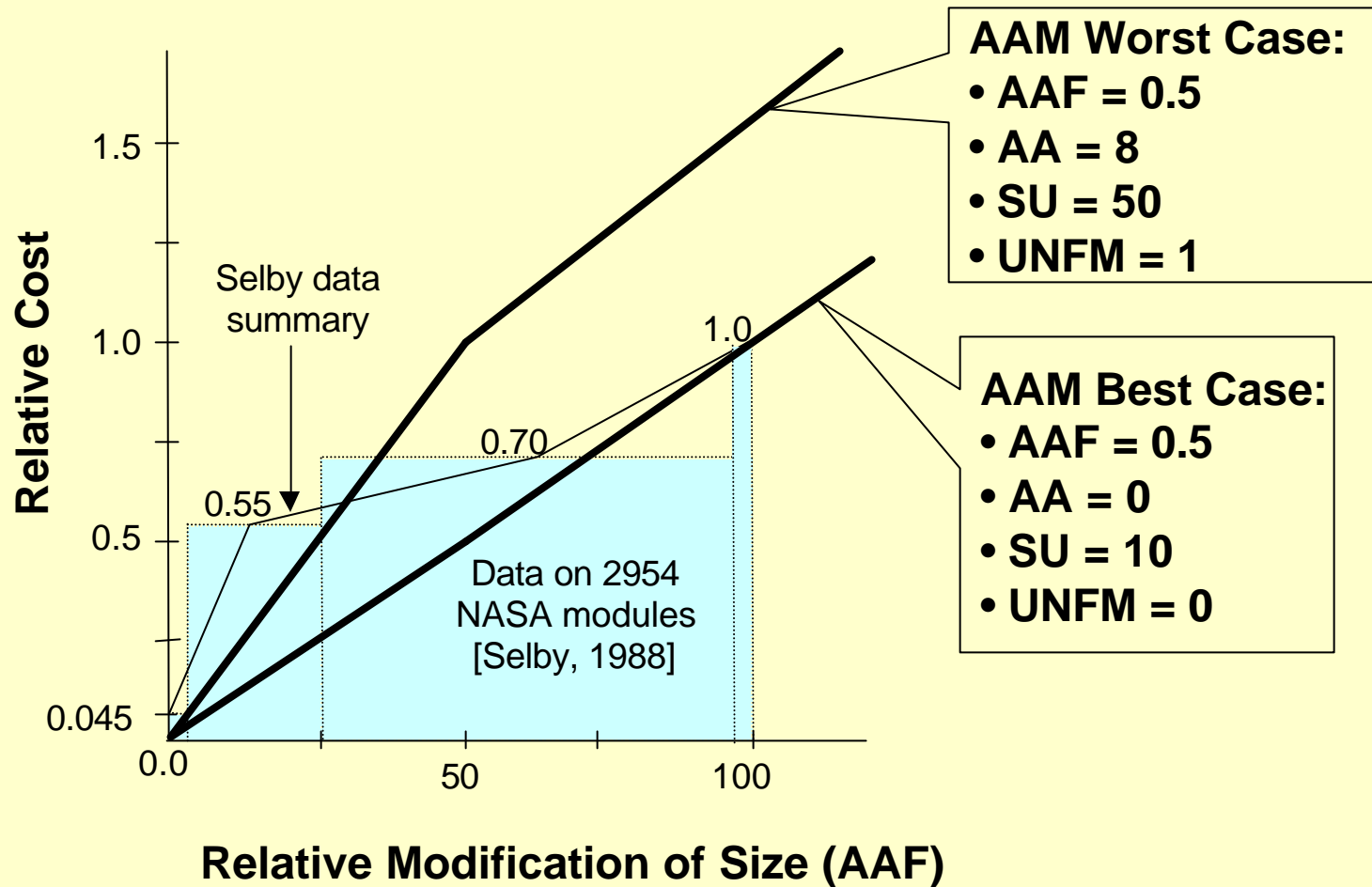
- **This is the reuse size model and its effects are shown on the next slide**
 - **AT is the percentage of code that is reengineered by automatic translation**
 - **The other factors are explained next**

$$\text{Equivalent KSLOC} = \text{Adapted KSLOC} \times \left(1 - \frac{\text{AT}}{100}\right) \times \text{AAM}$$

$$\text{AAM} = \begin{cases} \frac{[\text{AA} + \text{AAF}(1 + (0.02 \times \text{SU} \times \text{UNFM}))]}{100}, & \text{for } \text{AAF} \leq 50 \\ \frac{[\text{AA} + \text{AAF} + (\text{SU} \times \text{UNFM})]}{100}, & \text{for } \text{AAF} > 50 \end{cases}$$

$$\text{AAF} = (0.4 \times \text{DM}) + (0.3 \times \text{CM}) + (0.3 \times \text{IM})$$

Nonlinear Reuse Effects





Adaptation Adjustment Modifiers

- **AAF: Adaptation Adjustment Factor (COCOMO 81)**
 - Percent Design and Code Modified (DM and CM)
 - Percent Integration Required for Adapted Software (IM)
- **AA: Assessment and Assimilation increment (0 - 8)**
 - Determine whether a reused software module is appropriate to the application and to integrate its description into the overall product description
- **SU: Software Understanding increment (50 - 10)**
 - To cover nonlinear software understanding effects
 - Coupled with software unfamiliarity level (UNFM)
 - Apply only if reused software is modified
- **UNFM: Programmer Unfamiliarity**
 - Handles situations where the programmers are familiar with the software to be adapted.



SU Rating / Increment Example

	Very Low	Low	Nominal	High	Very High
Structure	Very low cohesion, high coupling, spaghetti code.	Moderately low cohesion, high coupling.	Reasonably well-structured; some weak areas.	High cohesion, low coupling.	Strong modularity, information hiding in data / control structures.
Application Clarity	No match between program and application world-views.	Some correlation between program and application.	Moderate correlation between program and application.	Good correlation between program and application.	Clear match between program and application world-views.
Self-Descriptiveness	Obscure code; documentation missing, obscure or obsolete	Some code commentary and headers; some useful documentation.	Moderate level of code commentary, headers, documentation.	Good code commentary and headers; useful documentation; some weak areas.	Self-descriptive code; documentation up-to-date, well-organized, with design rationale.
SU Increment to ESLOC	50	40	30	20	10



COCOMO II Model Overview

- COCOMO II Overview
- Sizing the Application
- ➔ • Estimating Effort
- Estimating Schedule
- Understanding model workings
- Estimating Software Maintenance



Early Design and Post-Arch Models (revisited)

- **Estimated Effort:**

$$PM = A \times (\text{Size})^E \times \prod_{i=1}^n EM_i$$

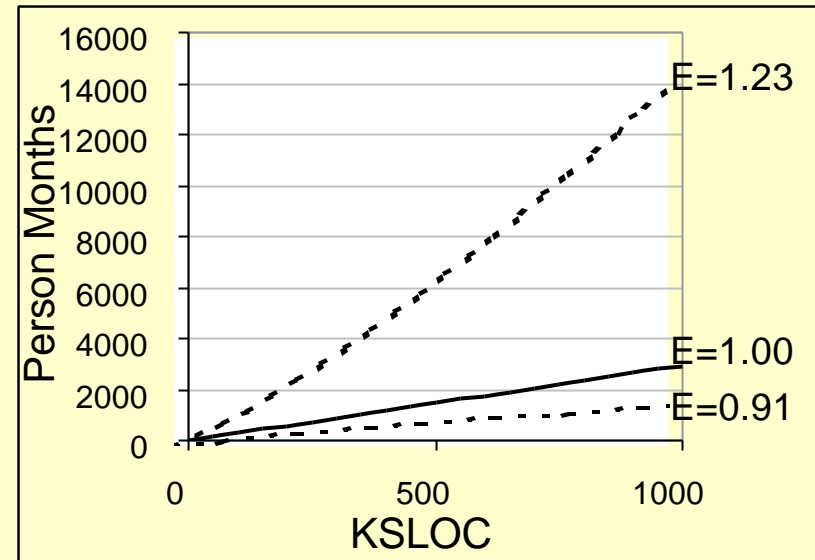
$$\text{where } E = B + 0.01 \times \sum_{j=1}^5 SF_j$$

- **A = 2.94** **B = 0.91**
- **Size (discussed earlier)**
- **EM: Effort Multipliers (7 for ED, 17 for PA)**
- **SF: Scale Factors (5 for both models)**

Project Scale Factors

$$E = B + 0.01 \times \sum_{j=1}^5 SF_j$$

- **E ranges from 0.91 to 1.23**
- **Apply to whole project**
 - Precedentedness
 - Development flexibility
 - Architecture/ risk resolution
 - Team cohesion
 - Process maturity (derived from SEI SW-CMM)





Project Scale Factors (cont.)

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
PREC	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar	largely familiar	thoroughly familiar
SF₁:	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
SF₂:	5.07	4.05	3.04	2.03	1.01	0.00
RESL	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
SF₃:	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions
SF₄:	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	SW-CMM Level 1 Lower	SW-CMM Level 1 Upper	SW-CMM Level 2	SW-CMM Level 3	SW-CMM Level 4	SW-CMM Level 5
SF₅:	7.80	6.24	4.68	3.12	1.56	0.00
EPML	or the Equivalent Process Maturity Level					
	5	4	3	2	1	0

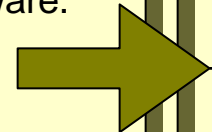
Equivalent Process Maturity Level

Requirements Management KPA: involves establishing and maintaining an agreement with the customer on the requirements for the software project.

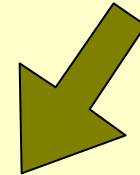
Goal 1: System requirements allocated to software are controlled to establish a baseline for software engineering and management use.

Goal 2: Software plans, products, and activities are kept consistent with the system requirements allocated to software.

- _ Almost Always (over 90% of the time)
- _ Frequently (about 60% to 90% of the time)
- _ About Half (about 40% to 60% of the time)
- _ Occasionally (about 10% to 40% of the time)
- _ Rarely if ever (less than 10% of the time)
- _ Does not apply
- _ Do not know



After each KPA is rated, the rating level is weighted:
100% for Almost Always
75% for Frequently
50% for About Half
25% for Occasionally
1% for Rarely If Ever).



$$EPML = 5 \times \left(\sum_{i=1}^n \frac{KPA\%_i}{100} \right) \cdot \frac{1}{n}$$



Cost Drivers

- **Cost drivers produce the EM_i values in the COCOMO II effort equation**
- **Post-Architecture Model has 17 cost drivers**
 - Product
 - Platform
 - Personnel
 - Project
- **Early Design Model has 7 cost drivers**
- **All cost drivers except Required Development Schedule can be applied to subsystems or modules**



Post-Arch Cost Drivers: Product

Cost Drivers	Very Low	Low	Nominal	High	Very High	Extra High
Required Reliability (RELY)	slight inconvenience	low, easily recoverable losses	moderate, easily recoverable losses	high financial loss	risk to human life	
Database Size (DATA)		DB bytes / Pgm SLOC < 10	$10 \leq D/P < 100$	$100 \leq D/P < 1000$	$D/P > 1000$	
Product Complexity (CPLX)		see Complexity Table				
Developed for Reusability (RUSE)		none	across project	across program	across product line	across multiple product lines
Documentation Match to LC Needs (DOCU)	Many life-cycle needs uncovered	Some life-cycle needs uncovered.	Right-sized to life-cycle needs	Excessive for life-cycle needs	Very excessive for life-cycle needs	



Post-Arch Cost Driver: Complexity

CPLX Ratings	Control Operations	Computational Operations	Device-dependent Operations	Data Management Operations	User Interface Management Operations
Very Low
Low
Nominal	Mostly simple nesting. Some intermodule control. Decision tables. Simple callbacks or message passing, including middleware-supported distributed processing	Use of standard math and statistical routines. Basic matrix/vector operations.	I/O processing includes device selection, status checking and error processing.	Multi-file input and single file output. Simple structural changes, simple edits. Complex COTS-DB queries, updates.	Simple use of widget set.
High
Very High
Extra High



Post-Arch Cost Drivers: Platform

Cost Drivers	Very Low	Low	Nominal	High	Very High	Extra High
Execution Time Constraint (TIME)			≤ 50% use of available execution time	70%	85%	95%
Main Storage Constraint (STOR)			≤ 50% use of available storage	70%	85%	95%
Platform Volatility (PVOL)		major change every 12 mo.; minor change every 1 mo.	major: 6 mo.; minor: 2 wk.	major: 2 mo.; minor: 1 wk.	major: 2 wk.; minor: 2 days	



Post-Arch Cost Drivers: Personnel

Cost Drivers	Very Low	Low	Nominal	High	Very High	Extra High
Analyst Capability (ACAP)	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	n/a
Programmer Capability (PCAP)	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	n/a
Personnel Continuity (PCON)	48% / year	24% / year	12% / year	6% / year	3% / year	n/a
Application Experience (APEX)	≤ 2 months	6 months	1 year	3 years	6 years	n/a
Platform Experience (PLEX)	≤ 2 months	6 months	1 year	3 years	6 year	n/a
Language and Tool Experience (LTEX)	≤ 2 months	6 months	1 year	3 years	6 year	n/a



Post-Arch Cost Drivers: Project

Cost Drivers	Very Low	Low	Nominal	High	Very High	Extra High
Use of Software Tools (TOOL)	edit, code, debug	simple, frontend, backend CASE, little integration	basic lifecycle tools, moderately integrated	strong, mature lifecycle tools, moderately integrated	strong, mature, proactive lifecycle tools, well integrated with processes, methods, reuse	
Multisite Development Collocation Communication (SITE)	International Some phone, mail	Multi-city and multi-company Individual phone, FAX	Multi-city or multi-company Narrow-band email	Same city or metro area Wide-band electronic communication.	Same building or complex Wide-band elect. comm, occasional video conf.	Fully collocated Interactive multimedia
Required Development Schedule (SCED)	75% of nominal	85% of nominal	100% of nominal	130% of nominal	160% of nominal	



Cost Driver: Required Development Schedule

- Required Development Schedule is the only cost driver that is **applied project wide**
- It appears in both the Post-Architecture and Early Design models
- *Nominal* is the duration estimate from $TDEV_{NS}$ (as shown in the opening example)

Cost Driver	Very Low	Low	Nominal	High	Very High	Extra High
Required Development Schedule (SCED)	75% of nominal	85% of nominal	100% of nominal	130% of nominal	160% of nominal	n/a



COCOMO II.2000 Post- Architecture Calibrated values

Baseline Effort Constants: A = 2.94; B = 0.91						
Baseline Schedule Constants: C = 3.67; D = 0.28						
Cost Driver	Very Low	Low	Nominal	High	Very High	Extra High
RELY	0.82	0.92	1.00	1.10	1.26	
DATA		0.90	1.00	1.14	1.28	
CPLX	0.73	0.87	1.00	1.17	1.34	1.74
RUSE		0.95	1.00	1.07	1.15	1.24
DOCU	0.81	0.91	1.00	1.11	1.23	
TIME			1.00	1.11	1.29	1.63
STOR			1.00	1.05	1.17	1.46
PVOL		0.87	1.00	1.15	1.30	



COCOMO II.2000 Post- Architecture Calibrated values

Cost Driver	Very Low	Low	Nominal	High	Very High	Extra High
ACAP	1.42	1.19	1.00	0.85	0.71	
PCAP	1.34	1.15	1.00	0.88	0.76	
PCON	1.29	1.12	1.00	0.90	0.81	
APEX	1.22	1.10	1.00	0.88	0.81	
PLEX	1.19	1.09	1.00	0.91	0.85	
LTEX	1.20	1.09	1.00	0.91	0.84	
TOOL	1.17	1.09	1.00	0.90	0.78	
SITE	1.22	1.09	1.00	0.93	0.86	0.80
SCED	1.43	1.14	1.00	1.00	1.00	

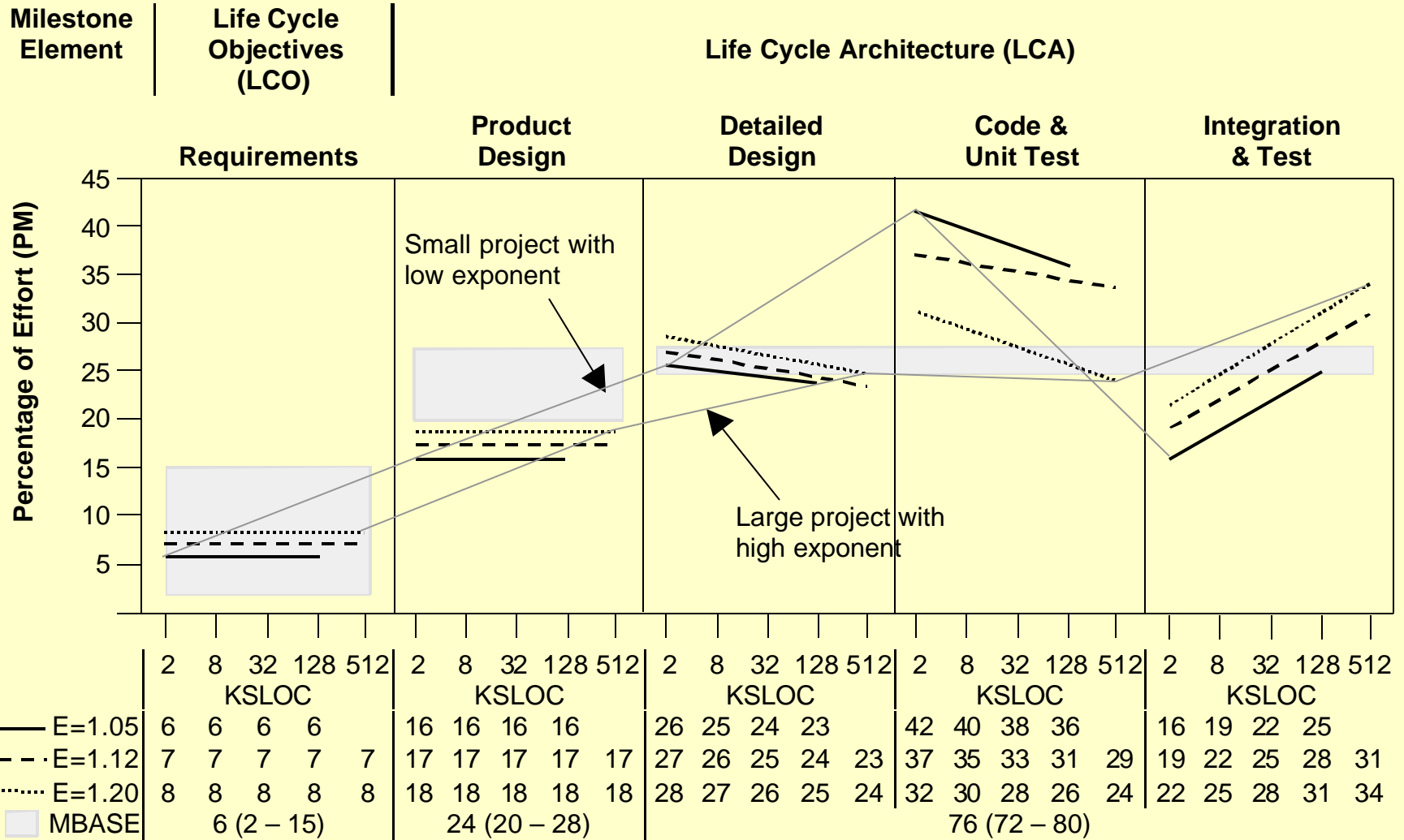


Early Design versus Post-Arch Cost Drivers

Early Design Cost Drivers	Counterpart Combined Post- Architecture Cost Drivers
Product Reliability and Complexity (RCPX)	RELY, DATA, CPLX, DOCU
Developed for Reusability (RUSE)	RUSE
Platform Difficulty (PDIF)	TIME, STOR, PVOL
Personnel Capability (PERS)	ACAP, PCAP, PCON
Personnel Experience (PREX)	APEX, PLEX, LTEX
Facilities (FCIL)	TOOL, SITE
Required Development Schedule (SCED)	SCED



Effort Distribution





Post-Arch Model Example (cont)

- $PM_{NS} = 2.94(100)^{(0.91+0.01*24)} = 586.61$ person months
 - $E = 1.15$

- **Effort Distribution:**

- Requirements: 41 PM

Code & Unit Test: 182 PM

- Product Design: 100 PM

Integration & Test: 164 PM

- Detailed Design: 141 PM



COCOMO II Model Overview

- **COCOMO II Overview**
- **Sizing the Application**
- **Estimating Effort**
- ➔ • **Estimating Schedule**
- **Understanding model workings**
- **Estimating Software Maintenance**



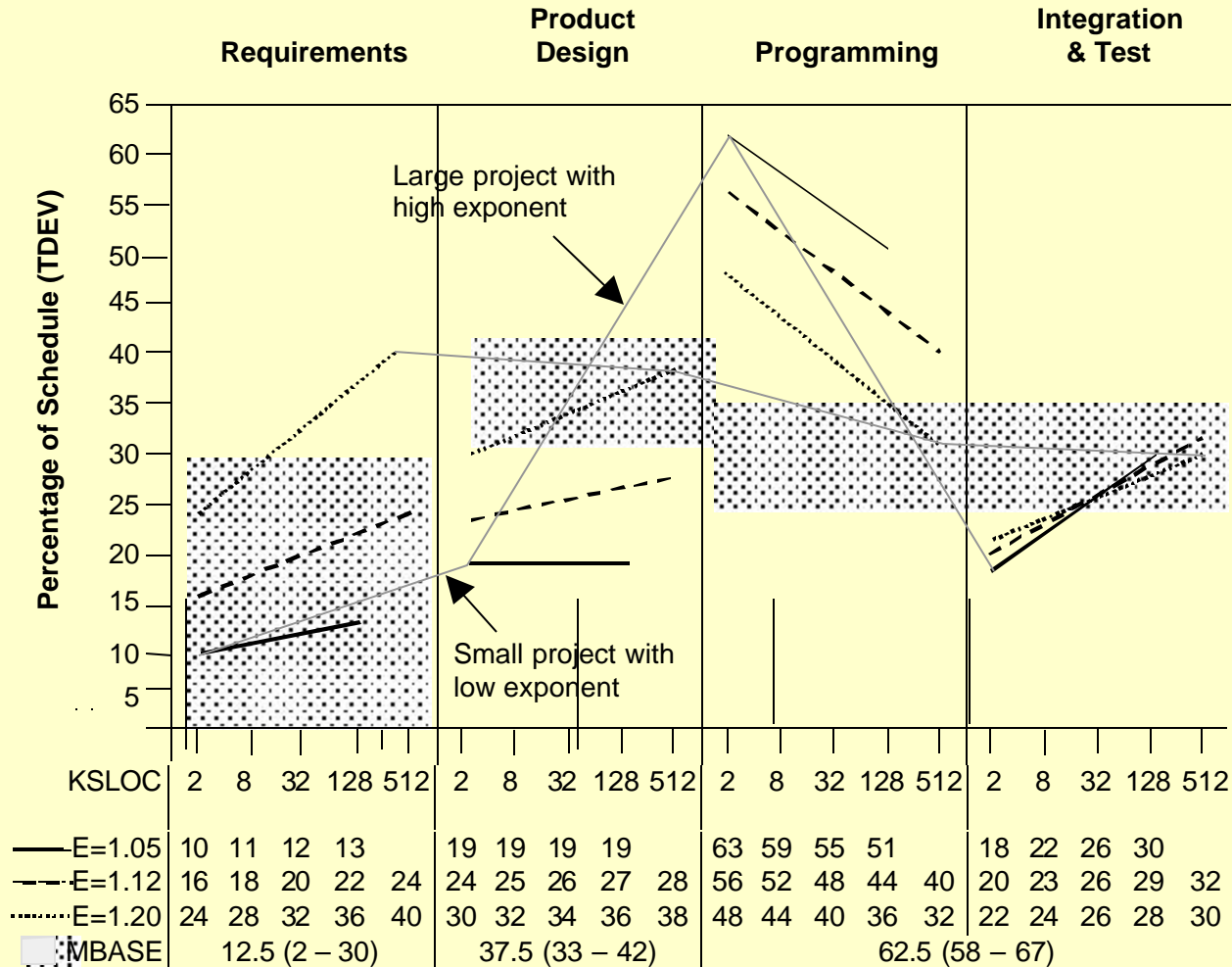
Early Design and Post-Arch Models (revisited)

- **Estimated Duration (schedule):**

$$\text{TDEV} = \left[C \times (\text{PM}_{\text{NS}})^{D+0.2 \times 0.1 \times \sum_{j=1}^5 \text{SF}_j} \right] \times \frac{\text{SCED}\%}{100}$$

- **TDEV: Time to Develop**
- **C = 3.67** **D = 0.28**
- **Where PM_{NS} is the estimated nominal-schedule person months (excludes SCED cost driver effects)**
- **SF: Scale Factors (5 for both models)**
- **SCED%: Percentage of schedule compression from nominal (from the SCED cost driver)**

Schedule Distribution



Distribution of Schedule by Size and Scale Factor



Post-Arch Model Example (cont)

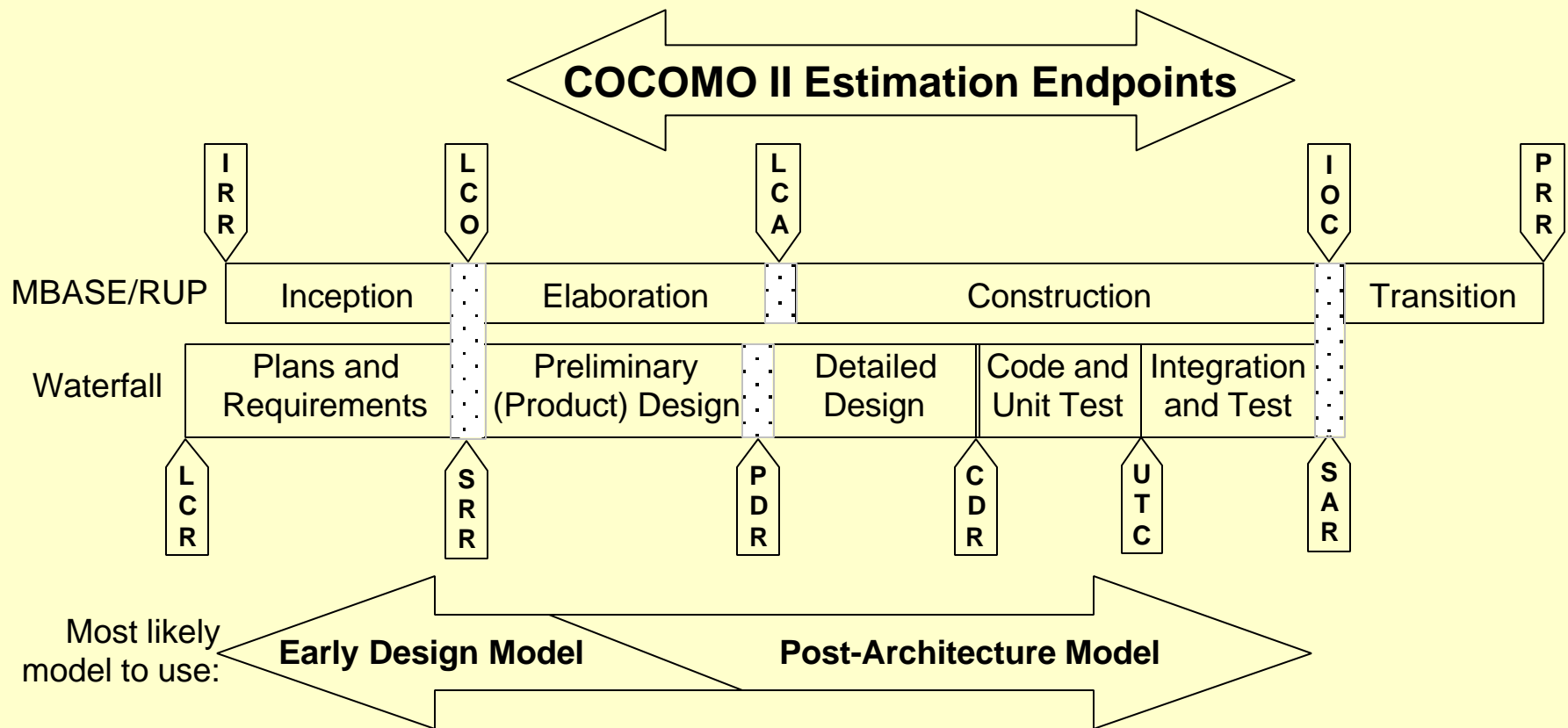
- $TDEV_{NS} = 3.67(586.6)^{(0.28+0.2*0.01*24)} = 29.7$ months
- **Schedule Distribution:**
 - Requirements: 5 Months
 - Product Design: 6 Months
 - Programming: 8 Months
 - Integration & Test: 6 Months



COCOMO II Model Overview

- COCOMO II Overview
- Sizing the Application
- Estimating Effort
- Estimating Schedule
- ➔ • Understanding model workings
- Estimating Software Maintenance

Model Breadth: Life Cycle Phases



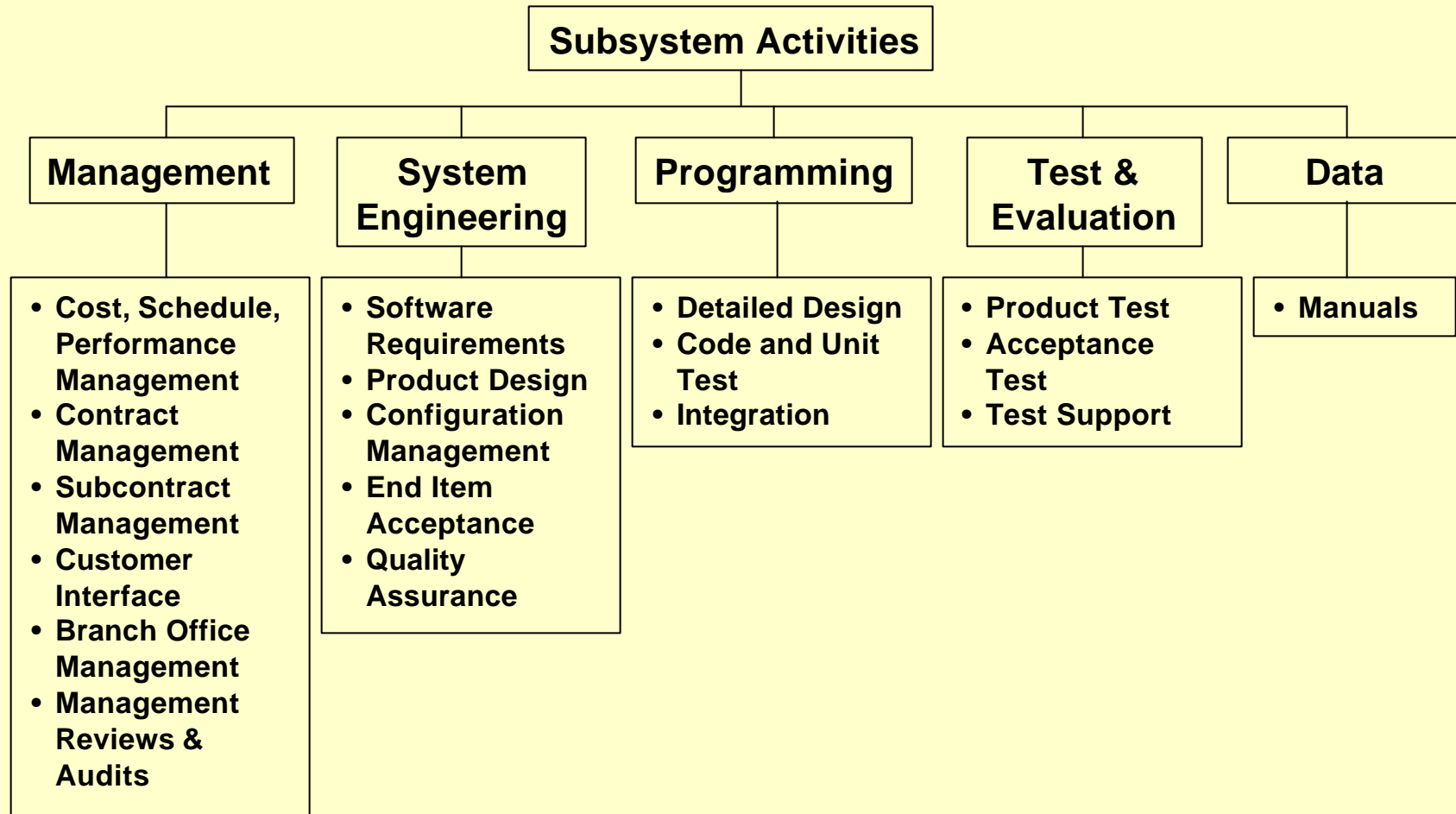


MBASE and RUP Phase Distribution

Phase (Endpoints)	MBASE		RUP	
	Effort%	Schedule%	Effort%	Schedule%
Inception (IRR to LCO)	6 (2-15)	12.5 (2-30)	5	10
Elaboration (LCO to LCA)	24 (20-28)	37.5 (33-42)	20	30
Construction (LCA to IOC)	76 (72-80)	62.5 (58-67)	65	50
Transition (IOC to PRR)	12 (0-20)	12.5 (0-20)	10	10
Totals:	118	125	100	100



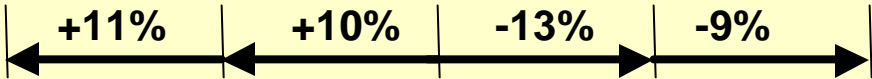
Model Depth: Activity WBS



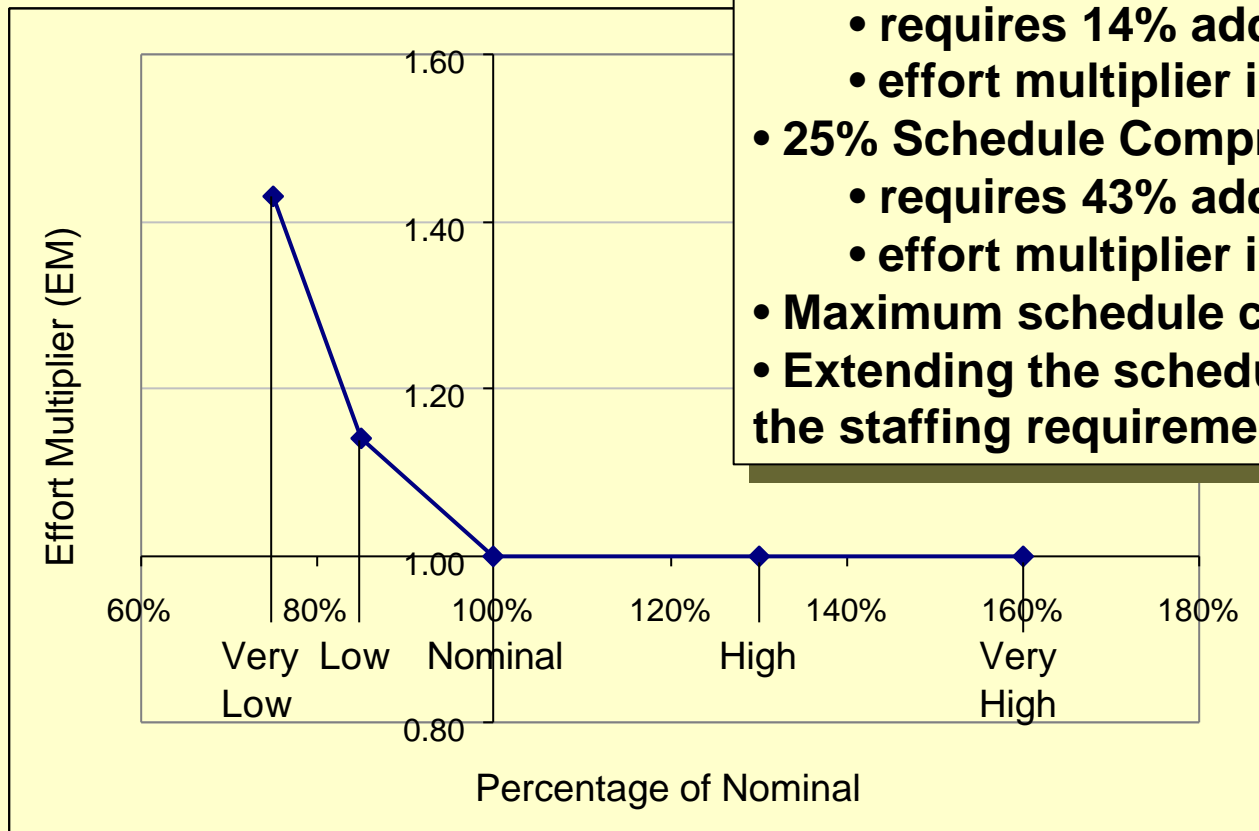
Impact of Application Experience

The level of applications experience of the project team developing the software system or subsystem. The ratings are defined in terms of the project team's equivalent level of experience with this type of application.

Descriptors:	2 ½ months	6 months	1 year	3 years	6 years
Rating Levels	Very Low	Low	Nominal	High	Very High
Effort Multipliers	1.22	1.10	1.00	0.88	0.81

Effect on Effort: 

Effects of Schedule Compression



- **15% Schedule Compression:**
 - requires 14% additional staff
 - effort multiplier increase 1.14
- **25% Schedule Compression:**
 - requires 43% additional staff
 - effort multiplier increase 1.43
- **Maximum schedule compression is 25%**
- **Extending the schedule does not change the staffing requirement**



Model Particulars

- **Size is for delivered product size**
- **A Person Month is 152 hours of effort**
- **Calibration is from the end of requirements analysis to the end of integration and test. Other phases are “add-ons”**
- **The model is more accurate if it is calibrated to local development conditions**



COCOMO II Model Overview

- **COCOMO II Overview**
- **Sizing the Application**
- **Estimating Effort**
- **Estimating Schedule**
- **Understanding model workings**
- ➔ • **Estimating Software Maintenance**



Sizing Software Maintenance -1

- **Apply Scale Factors, E, to the size of code being changed (rather than the whole product being modified)**
- **Account for effects of modification to an existing body of code with software understanding (SU) and programmer unfamiliarity (UNFM)**
- **Excludes**
 - **Major product rebuilds (over 50%)**
 - **Development of sizeable interfaces (over 20%)**



Sizing Software Maintenance -2

$$\text{Size}_M = [(\text{Base Code Size}) \times \text{MCF}] \times \text{MAF}$$

$$\text{where MCF} = \frac{(\text{Size Added}) + (\text{Size Modified})}{(\text{Base Code Size})}$$

$$\text{where MAF} = 1 + \left(\frac{\text{SU}}{100} \times \text{UNFM} \right)$$

- **MCF:** Maintenance Change Factor is the percentage of change to the base code
- **MAF:** Maintenance Adjustment Factor is used to adjust the effective maintenance size to account for software understanding and programmer familiarity with the software being maintained



Maintenance Considerations

- **SCED cost driver is not used - maintenance cycle is considered fixed**
- **RUSE cost driver is not used - maintaining a component is balanced by is careful design, documentation and testing**
- **RELY cost driver has a different set of effort multipliers (EM). Depends on the required reliability under which the product was developed**



Maintenance Model

- **Maintenance effort and schedule estimation:**

$$PM_M = A \times (\text{Size}_M)^E \times \prod_{i=1}^{15} EM_i$$

$$FSPM = \frac{PM_M}{TM}$$

- **A = 2.94, E = scaling exponent**
- **Size_M = size of code being changed**
- **TM = Fixed time for maintenance**
- **FSPM = Full Time Software Personnel for Maintenance**



Questions?

Software Cost Estimation with COCOMO II

Barry W. Boehm
Chris Abts
A. Winsor Brown
Sunita Chulani
Bradford K. Clark
Ellis Horowitz
Ray Madachy
Donald Reifer
Bert Steece

Printice Hall PTR
Upper Saddle River, NJ