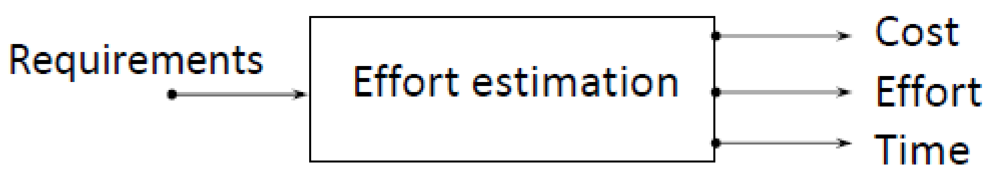
7. Effort estimation

How to evaluate the effort in order to dev a Software



Requirements: functionalities, quality etc.

Steps to follow

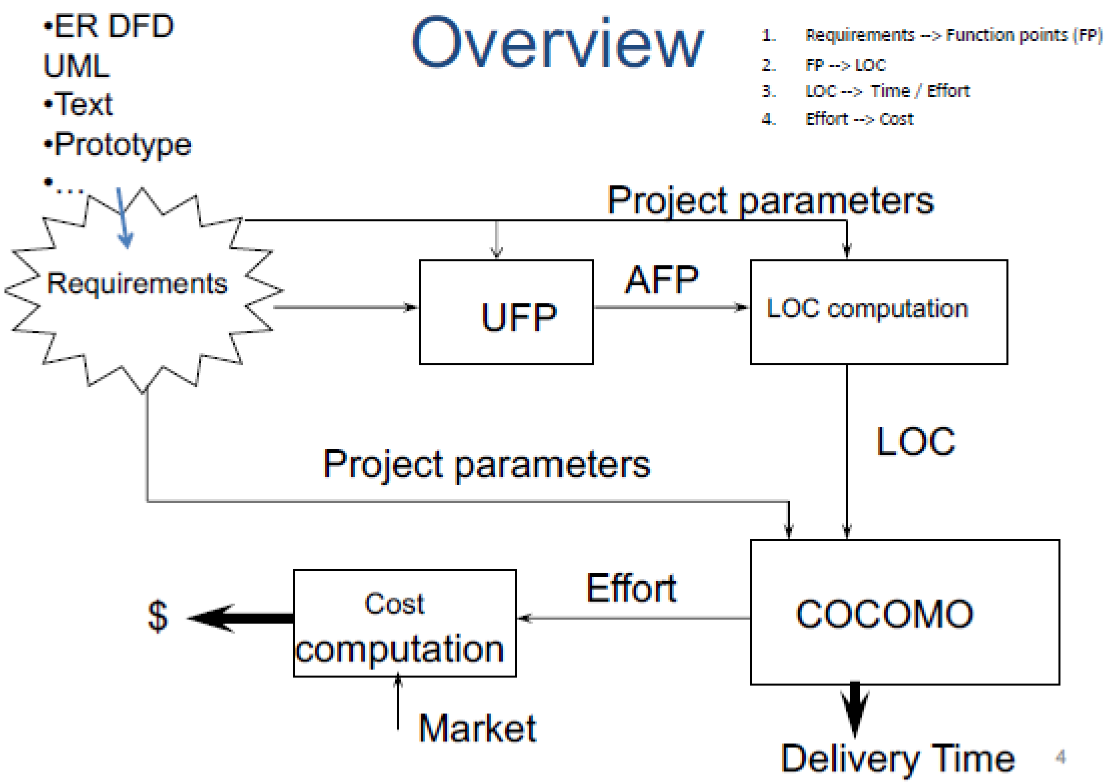
Requirements → FP

FP → LOC

LOC → Time/Effort

Effort → Cost

7.1 CoCoMo



UFP: Unadjasted FP → AFP

Project par: are the type of project I want to follow (approach, technology, paradigm etc.)

LOC + Project parameters → COCOMO:

is the technique that is able to estimate effort and delivery time of a project.

Some statistics

	Project dimension			
	<100	100-1K	1K-5K	>5K
Deleted	3	7	13	24
1 year delay	1	10	12	18
6-12 month delay	9	24	35	37
<6 month delay	72	53	37	20
Early completion	15	6	3	1

	Project dimension			
Duration	<100	100-1K	1K-5K	>5K
Planned	6	12	18	24
Real	6	16	24	36
Delay	0	4	6	12

Effort estimation from LOC

based on CoCoMo: Contructive Cost Model

Estimates effort M and optimal T (time), relies on statistics, waterfall model (!), basic formula:

$$M=aS^b \quad T=cM^d \text{ (S represents KLOC)}$$

Provide effort indication in:

- Analysis and planning
- design
- dev
- testing

M: Effort, cost: Man time required (man-day, man-month)

T: Delivery time: Required (optimal) time (years, months)

Manpower: number of people working on the project

CoCoMo Formulas:

Tipo di modello	Base	Intermedio	Dettagliato
Caratteristiche generali progetto	Solo dimensione complessiva	coefficienti di correzione globali	coefficienti di correzione per ciascuna fase
Semplice (organic)	$M = 2{,}4\, S_k^{1,05}$ $T = 2{,}5\, M^{0,38}$	$M = M_{Nom} \prod_1^{15} c_i$ $M_{Nom} = 3{,}2\, S_k^{1,05}$	idem
Intermedio (semi-detached)	$M = 3{,}0\, S_k^{1,12}$ $T = 2{,}5\, M^{0,35}$	$M = M_{Nom} \prod_1^{15} c_i$ $M_{Nom} = 3{,}0\, S_k^{1,12}$	idem
Complesso (embedded)	$M = 3{,}6\, S_k^{1,20}$ $T = 2{,}5\, M^{0,32}$	$M = M_{Nom} \prod_1^{15} c_i$ $M_{Nom} = 2{,}8\, S_k^{1,20}$	idem

S: onyl lines of code developed

T: design-coding-integration and test (not reqs anal and planning)

MM: Man-month (152 hour)

See example p.14

M distribution % (organic)

KLOC

	2	8	32	128
Requirement analysis and planning	6	6	6	6
Design	16	16	16	16
Development	68	65	62	59
Detailed design	26	25	24	23
Coding & testing	42	40	38	36
Integration and test	16	19	22	25

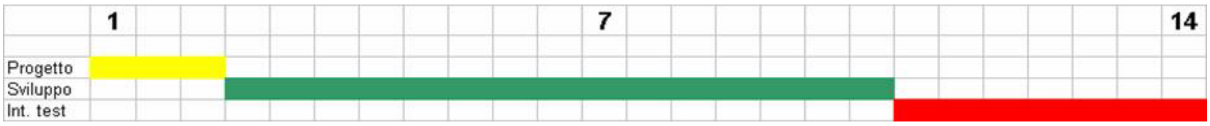
T distribution %

KLOC

	2	8	32	128
Requirement analysis and planning	10	11	12	13
Design	19	19	19	19
Development	63	59	55	51
Integration and test	18	22	26	30

Gantt

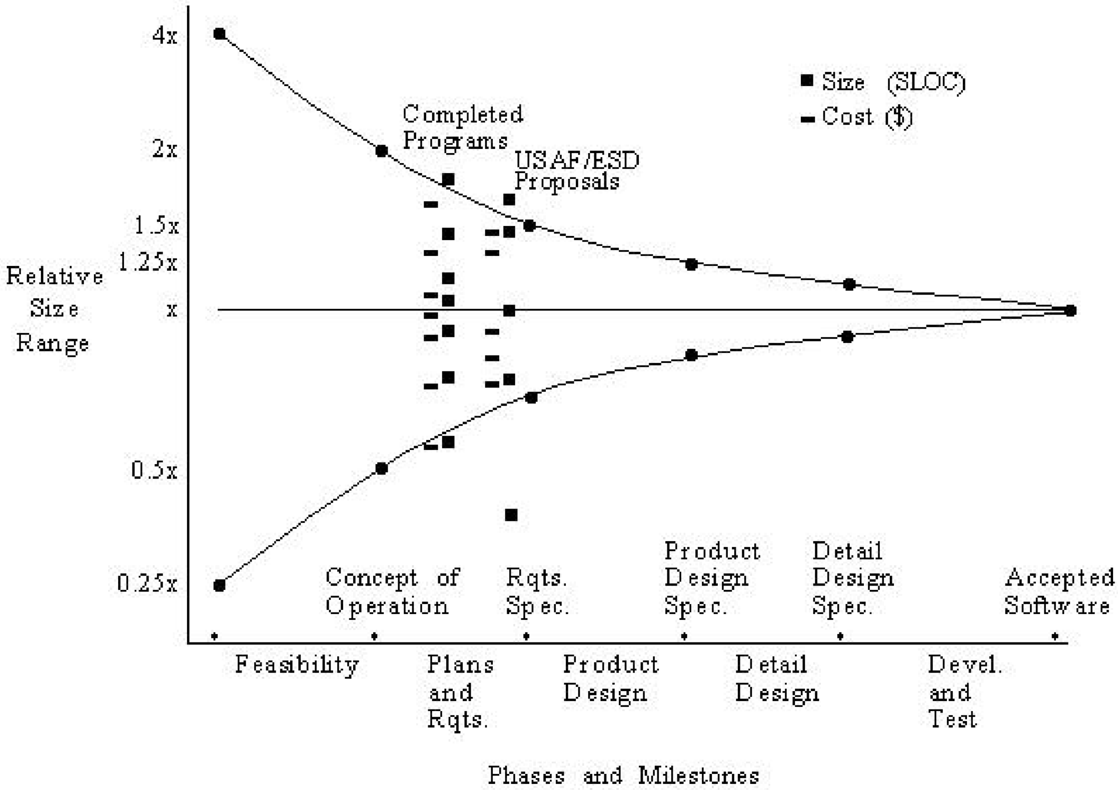
division in months;



7.2 CoCoMo II

Motivation: new lifecycle sw models, reuse, different leveles of estimation precision

Estimation precision



Models:

- Early design model (it must not be too much detailed)
- Post-architecture model (for maintennce and dev)

Cocomo II formulas

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

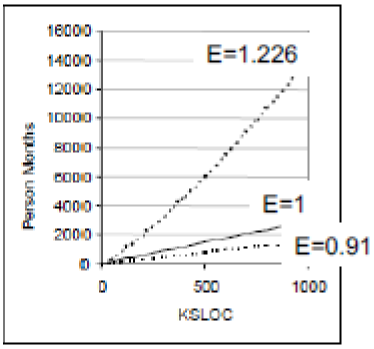
n=6+SCED or n=16+SCED

$$\left(\prod_{i=1} EM_i \right)$$

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

TDEV=C(PM_{NS})^{F} SCED/100

F = D+0.2(E-B)



- A, B, C, D -> constants
- SCED modify nominal schedule
- **E** denotes economy and diseconomy scales
- In Cocomo 1981 E={1.05, 1.12, 1.20} (only diseconomies)
- In Cocomo II: E= ranges between 0.91 a 1.226

Cocomo.24

Actual calibration

A=2.94

B=0.91

C=3.67

D=0.28

PM= **2.94** $S^{E} \times \prod_{i=1}^{n} EM_i$

E=**0.91**$+ 0.01 \times \sum_{j=1}^{5} SF_j$

TDEV = **3.67** ($PM_{adjusted}$)^{F} SCED%/100

F = **0.28** +0.2 x $0.01 \times \sum_{j=1}^{5} SF_j$

Scale factors

Table 10. Scale Factor Values, SF_j, for COCOMO II Models

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
PREC	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar	largely familiar	thoroughly familiar
SF_j:	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
SF_j:	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%)
SF_j:	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
SF_j:	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	The estimated Equivalent Process Maturity Level (EPML) or					
	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
SF_j:	7.80	6.24	4.68	3.12	1.56	0.00

PRECedenteness & Development **FLEX**ibility

Feature	Very Low	Nominal / High	Extra High
Precedentedness			
Organizational understanding of product objectives	General	Considerable	Thorough
Experience in working with related software systems	Moderate	Considerable	Extensive
Concurrent development of associated new hardware and operational procedures	Extensive	Moderate	Some
Need for innovative data processing architectures, algorithms	Considerable	Some	Minimal
Development Flexibility			
Need for software conformance with pre-established requirements	Full	Considerable	Basic
Need for software conformance with external interface specifications	Full	Considerable	Basic
Premium on early completion	High	Medium	Low

The PREC and FLEX scale factors are largely **intrinsic** to a project and **uncontrollable**

Architecture/Risk **RESoL**ution

Product Design Review

Component Off The Shelf

Characteristic	Very Low	Low	Nominal	High	Very High	Extra High
Risk Management Plan identifies all critical risk items, establishes milestones for resolving them by PDR.	None	Little	Some	Generally	Mostly	Fully
Schedule, budget, and internal milestones through PDR compatible with Risk Management Plan	None	Little	Some	Generally	Mostly	Fully
Percent of development schedule devoted to establishing architecture, given general product objectives	5	10	17	25	33	40
Percent of required top software architects available to project	20	40	60	80	100	120
Tool support available for resolving risk items, developing and verifying architectural specs	None	Little	Some	Good	Strong	Full
Level of uncertainty in Key architecture drivers: mission, user interface, COTS, hardware, technology, performance.	Extreme	Significant	Consider-able	Some	Little	Very Little
Number and criticality of risk items	> 10 Critical	5-10 Critical	2-4 Critical	1 Critical	> 5Non-Critical	< 5 Non-Critical

TEAM Cohesion

Characteristic	Very Low	Low	Nominal	High	Very High	Extra High
Consistency of stakeholder objectives and cultures	Little	Some	Basic	Consider-able	Strong	Full
Ability, willingness of stakeholders to accommodate other stakeholders' objectives	Little	Some	Basic	Consider-able	Strong	Full
Experience of stakeholders in operating as a team	None	Little	Little	Basic	Consider-able	Extensive
Stakeholder teambuilding to achieve shared vision and commitments	None	Little	Little	Basic	Consider-able	Extensive

Process MATurity

- Based on CMMI
- Two calculation methods:
  - CMMI level (1-, 1+, 2, 3, 4 , 5)
  - Implementation % of the 18 key process areas

$$5 - \left[\sum_{i=1}^{18} \left(\frac{KPA\%_i}{100} \times \frac{5}{18} \right) \right]$$

Key Process Areas	Almost Always (>90%)	Often (60-90%)	About Half (40-60%)	Occasion-ally (10-40%)	Rarely If Ever (<10%)	Does Not Apply	Don't Know
Requirements Management	☐	☐	☐	☐	☐	☐	☐
Software Project Planning	☐	☐	☐	☐	☐	☐	☐
Software Project Tracking and Oversight	☐	☐	☐	☐	☐	☐	☐
Software Subcontract Management	☐	☐	☐	☐	☐	☐	☐
Software Quality Assurance	☐	☐	☐	☐	☐	☐	☐
Software Configuration Management	☐	☐	☐	☐	☐	☐	☐
Organization Process Focus	☐	☐	☐	☐	☐	☐	☐
Organization Process Definition	☐	☐	☐	☐	☐	☐	☐
Training Program	☐	☐	☐	☐	☐	☐	☐
Integrated Software Management	☐	☐	☐	☐	☐	☐	☐
Software Product Engineering	☐	☐	☐	☐	☐	☐	☐
Intergroup Coordination	☐	☐	☐	☐	☐	☐	☐
Peer Reviews	☐	☐	☐	☐	☐	☐	☐
Quantitative Process Management	☐	☐	☐	☐	☐	☐	☐
Software Quality Management	☐	☐	☐	☐	☐	☐	☐
Defect Prevention	☐	☐	☐	☐	☐	☐	☐
Technology Change Management	☐	☐	☐	☐	☐	☐	☐
Process Change Management	☐	☐	☐	☐	☐	☐	☐

Effort multipliers

Post-Architecture model	$PM = \mathbf{2.94}\ S^{E} \times \left(\prod_{i=1}^{17} EM_i\right)$

Early Design model	$PM = \mathbf{2.94}\ S^{E} \times \left(\prod_{i=1}^{7} EM_i\right)$

SCED + Effort multipliers

Product

- RELY: REquired software reliabiLitY(0.82-0.92-1.0-1.10-1.26-n/a)
- DATA: DATA base size (n/a-0.90-1.0-1.14-1.28-n/a)
- CPLX: product ComPLeXity (0.73-0.87-1.0-1.17-1.34-1.75)
- RUSE: intended reuse of product modules (n/a-0.95-1.0-1.07-1.15-1.24)
- DOCU: level of required documentation (0.81-0.91-1.0-1.11-1.23-n/a)

System

- TIME: execution TIME constraint (n/a-n/a-1.0-1.11-1.29-1.63)
- STOR: main STORage constraint (n/a-n/a-1.0-1.05-1.17-1.46)
- PVOL - Platform volatility (n/a-0.87-1.0-1.15-1.30-n/a)

Personal

- ACAP - Analyst CAPability (1.42-1.19-1.0-0.85-0.71-n/a)
- PCAP - Programmer CAPability (1.34-1.15-1.0-0.88-0.76-n/a)
- APEX - Application EXPerience (1.22-1.10-1.0-0.88-0.81-n/a)
- PLEX – Platform EXPerience (1.19-1.09-1.0-0.91-0.85-n/a)
- LTEX: Language and tool EXPerience (1.20-1.09-1.0-0.91-0.84-n/a)
- PCON:Personnel continuity(1.29-1.12-1.0-0.90-0.81-n/a)

Project

- SITE: MultiSITE development (1.22-1.09-1.0-0.93-0.86-0.80)
- TOOL - use of software TOOLs (1.17-1.09-1.0-0.90-0.78-n/a)
- SCED - SChEDule constraints (1.43-1.14-1.0-1.00-1.00-n/a)

Extended:

Data:

This cost driver attempts to capture the effect large test data requirements have on product development. The rating is determined by calculating D/P, the ratio of bytes in the testing database to SLOC in the program. The reason the size of the database is important to consider is because of the effort required to generate the test data that will be used to exercise the program. In other words, DATA is capturing the effort needed to assemble and maintain the data required to complete test of the program through IOC, see Table 18.

Table 18. DATA Cost Driver

DATA* Descriptors		Testing DB bytes/Pgm SLOC < 10	$10 \leq D/P < 100$	$100 \leq D/P < 1000$	$D/P \geq 1000$	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	0.90	1.00	1.14	1.28	n/a

* DATA is rated as Low if D/P is less than 10 and it is very high if it is greater than 1000. P is measured in equivalent source lines of code (SLOC), which may involve function point or reuse conversions.

CPLX:

Complexity is divided into five areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations. Using Table 19, select the area or combination of areas that characterize the product or the component of the product you are rating. The complexity rating is the subjective weighted average of the selected area ratings. Table 20 provides the COCOMO II.2000 effort multipliers for CPLX.

	Control Operations	Computational Operations	Device-dependent Operations	Data Management Operations	User Interface Management Operations
Very Low	Straight-line code with a few non-nested structured programming operators: DOs, CASEs, IF-THEN-ELSEs. Simple module composition via procedure calls or simple scripts.	Evaluation of simple expressions: e.g., A=B+C*(D-E)	Simple read, write statements with simple formats.	Simple arrays in main memory. Simple COTS-DB queries, updates.	Simple input forms, report generators.
Low	Straightforward nesting of structured programming operators. Mostly simple predicates	Evaluation of moderate-level expressions: e.g., D=SQRT(B**2-4.*A*C)	No cognizance needed of particular processor or I/O device characteristics. I/O done at GET/PUT level.	Single file subsetting with no data structure changes, no edits, no intermediate files. Moderately complex COTS-DB queries, updates.	Use of simple graphic user interface (GUI) builders.
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	Highly nested structured programming operators with many compound predicates. Queue and stack control. Homogeneous, distributed processing. Single processor soft real-time control.	Basic numerical analysis: multivariate interpolation, ordinary differential equations. Basic truncation, round-off concerns.	Operations at physical I/O level (physical storage address translations; seeks, reads, etc.). Optimized I/O overlap.	Simple triggers activated by data stream contents. Complex data restructuring.	Widget set development and extension. Simple voice I/O, multimedia.
Very High	Reentrant and recursive coding. Fixed-priority interrupt handling. Task synchronization, complex callbacks, heterogeneous distributed processing. Single-processor hard real-time control.	Difficult but structured numerical analysis: near-singular matrix equations, partial differential equations. Simple parallelization.	Routines for interrupt diagnosis, servicing, masking. Communication line handling. Performance-intensive embedded systems.	Distributed database coordination. Complex triggers. Search optimization.	Moderately complex 2D/3D, dynamic graphics, multimedia.
Extra High	Multiple resource scheduling with dynamically changing priorities. Microcode-level control. Distributed hard real-time control.	Difficult and unstructured numerical analysis: highly accurate analysis of noisy, stochastic data. Complex parallelization.	Device timing-dependent coding, micro-programmed operations. Performance-critical embedded systems.	Highly coupled, dynamic relational and object structures. Natural language data management.	Complex multimedia, virtual reality, natural language interface.

RUSE:

This cost driver accounts for the additional effort needed to construct components intended for reuse on the current or future projects. This effort is consumed with creating more generic design of software, more elaborate documentation, and more extensive testing to ensure components are ready for use in other applications.

	Very Low	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
RUSE		none	across project	across pro gram	across prod uct line	across mul tiple prod uct lines

DOCU:

	Very Low	Low	Nominal	High	Very High	Extra High
DOCU	Many life-cycle needs uncovered	Some life-cycle needs uncovered.	Right-sized to life-cycle needs	Excessive for life-cycle needs	Very exces sive for life-cycle needs	

TIME:

This is a measure of the execution time constraint imposed upon a software system. The rating is expressed in terms of the percentage of available execution time expected to be used by the system or subsystem consuming the execution time resource. The rating ranges from nominal, less than 50% of the execution time resource used, to extra high, 95% of the execution time resource is consumed.

	Very Low	Low	Nominal	High	Very High	Extra High
TIME			≤ 50% use of available execution time	70%	85%	95%

STOR:

This rating represents the degree of main storage constraint imposed on a software system or subsystem. Given the remarkable increase in available processor execution time and main storage, one can question whether these constraint variables are still relevant. However, many applications continue to expand to consume whatever resources are available, making these cost drivers still relevant. The rating ranges from nominal, less that 50%, to extra high, 95%.

	Very Low	Low	Nominal	High	Very High	Extra High
STOR			≤ 50% use of available storage	70%	85%	95%

PVOL:

"Platform" is used here to mean the complex of hardware and software (OS, DBMS, etc.) the software product calls on to perform its tasks. If the software to be developed is an operating system then the platform is the computer hardware. If a database management system is to be developed then the platform is the hardware and the operating system. If a network text browser is to be developed then the platform is the network, computer hardware, the operating system, and the distributed information repositories. The platform includes any compilers or assemblers supporting the development of the software system. This rating ranges from low, where there is a major change every 12 months, to very high, where there is a major change every two weeks.

	Very Low	Low	Nominal	High	Very High	Extra High
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	

PCAP:

	Very Low	Low	Nominal	High	Very High	Extra High
ACAP	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	

LTEX:

	Very Low	Low	Nominal	High	Very High	Extra High
AEXP	2 months	6 months	1 year	3 years	6 years	

PCON:

The rating scale for PCON is in terms of the project's annual personnel turnover: from 3%, very high, to 48%, very low.

	Very Low	Low	Nominal	High	Very High	Extra High
PCON	48% / year	24% / year	12% / year	6% / year	3% / year	

TOOL:

Software tools have improved significantly since the 1970's projects used to calibrate COCOMO. The tool rating ranges from simple edit and code, very low, to integrated lifecycle management tools, very high.

	Very Low	Low	Nominal	High	Very High	Extra High
TOOL	edit, code, debug	simple, fron tend, back end CASE, little integra tion	basic lifecy cle tools, moderately integrated	strong, mature life cycle tools, moderately integrated	strong, mature, pro active lifecy cle tools, well inte grated with processes, methods, reuse	

SITE:

SITE: Collocation Descriptors:	Inter-national	Multi-city and Multi-company	Multi-city or Multi-company	Same city or metro. area	Same building or complex	Fully collocated
SITE: Communications Descriptors:	Some phone, mail	Individual phone, FAX	Narrow band email	Wideband electronic communication.	Wideband elect. comm., occasional video conf.	Interactive multimedia
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.22	1.09	1.00	0.93	0.86	0.80

SCED:

This rating measures the schedule constraint imposed on the project team developing the software. The ratings are defined in terms of the percentage of schedule stretch-out or acceleration with respect to a nominal schedule for a project requiring a given amount of effort. Accelerated schedules tend to produce more effort in the earlier phases to eliminate risks and refine the architecture, more effort in the later phases to accomplish more testing and documentation in parallel. In Table 34, schedule compression of 75% is rated very low. A schedule stretch-out of 160% is rated very high. Stretch-outs do not add or decrease effort. Their savings because of smaller team size are generally balanced by the need to carry project administrative functions over a longer period of time. The nature of this balance is undergoing further research in concert with our emerging CORADMO extension to address rapid application development (goto http://sunset.usc.edu/COCOMOII/suite.html for more information).

SCED is the only cost driver that is used to describe the effect of schedule compression / expansion *for the whole project*. The scale factors are also used to describe the whole project. All of the other cost drivers are used to describe each module in a multiple module project. Using the COCOMO II Post-Architecture model for multiple module estimation is explained in Section 3.3.

Table 34. SCED Cost Driver

SCED Descriptors	75% of nominal	85% of nominal	100% of nominal	130% of nominal	160% of nominal	
Rating Level	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multiplier	1.43	1.14	1.00	1.00	1.00	n/a

SCED is also handled differently in the COCOMO II estimation of time to develop, TDEV. This special use of SCED is explained in Section 4.

Summary

	Very Low	Low	Nominal	High	Very High	Extra High
RELY	slight inconvenience	low, easily recoverable losses	Moderate, easily recoverable losses	high financial loss	risk to human life	
DATA		DB bytes/Pgm SLOC < 10	10 ≤ D/P < 100	100 ≤ D/P < 1000	D/P ≥ 1000	
CPLX	see Table II-15					
RUSE		none	Across project	across program	across product line	across multiple product lines
DOCU	Many life-cycle needs uncovered	Some life-cycle needs uncovered.	Right-sized to life-cycle needs	Excessive for life-cycle needs	Very exces sive for life-cycle needs	
TIME			50% use of available exe cution time	70%	85%	95%
STOR			50% use of available stor age	70%	85%	95%
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	
ACAP	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
PCAP	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
PCON	48% / year	24% / year	12% / year	6% / year	3% / year	
AEXP	≤ 2 months	6 months	1 year	3 years	6 years	
PEXP	≤ 2 months	6 months	1 year	3 years	6 year	
LTEX	≤ 2 months	6 months	1 year	3 years	6 year	
TOOL	edit, code, debug	simple, frontend, backend CASE, little integration	basic lifecycle tools, moder ately inte grated	strong, mature lifecycle tools, moderately integrated	strong, mature, proactive life cycle tools, well inte grated with processes, methods, reuse	
SITE: Collocation	International	Multi-city and Multi-com pany	Multi-city or Multi-com pany	Same city or metro. area	Same building or complex	Fully collo cated
SITE: Communications	Some phone, mail	Individual phone, FAX	Narrowband email	Wideband electronic communica tion.	Wideband elect. comm, occasional video conf.	Interactive multimedia
SCED	75% of nominal	85%	100%	130%	160%	

Variability:

Best case: 0.09

Worst case: 115

7 vs 17

Early Design Cost Driver	Counterpart Combined Post-Architecture Cost Drivers
RCPX	RELY, DATA, CPLX, DOCU
RUSE	RUSE
PDIF	TIME, STOR, PVOL
PERS	ACAP, PCAP, PCON
PREX	AEXP, PEXP, LTEX
FCIL	TOOL, SITE
SCED	SCED

PM and T_{dev}

$$PM = \mathbf{2.94}\ S^{E} \times \left(\prod_{i=1}^{17} EM_i \right)$$

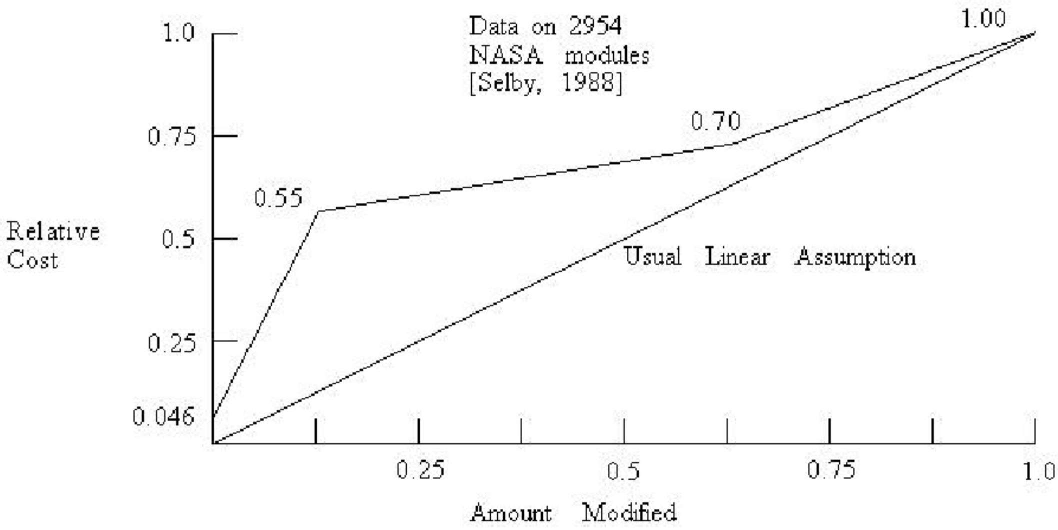
SCED

$$PM = \mathbf{2.94}\ S^{E} \times \left(\prod_{i=1}^{7} EM_i \right)$$

A=2.94
B=0.91
C=3.67
D=0.28

$$TDEV = [C \times (PM_{NS})^{(D+0.2\times(E-B))}] \times \frac{SCED\%}{100}$$

Reuse: how to estimate the effort for reusing modules



model the effort to adapt an existing module.

non linear model:

1) inherent complexity of adapting sw:

SU : Software Understanding

	Very Low	Low	Nom	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, documentations.	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

Penality percentage

AA: Assessment and Assimilation

AA Increment	Level of AA Effort
0	None
2	Basic module search and documentation
4	Some module Test and Evaluation (T&E), documentation
6	Considerable module T&E, documentation
8	Extensive module T&E, documentation

UNFM: Programmer Unfamiliarity

programmer's relative unfamiliarity with the software

UNFM Increment	Level of Unfamiliarity
0.0	Completely familiar
0.2	Mostly familiar
0.4	Somewhat familiar
0.6	Considerably familiar
0.8	Mostly unfamiliar
1.0	Completely unfamiliar

2) Percentage of modification

Percentage of modified design, modified code, modification to original integration

Equivalent SLOC

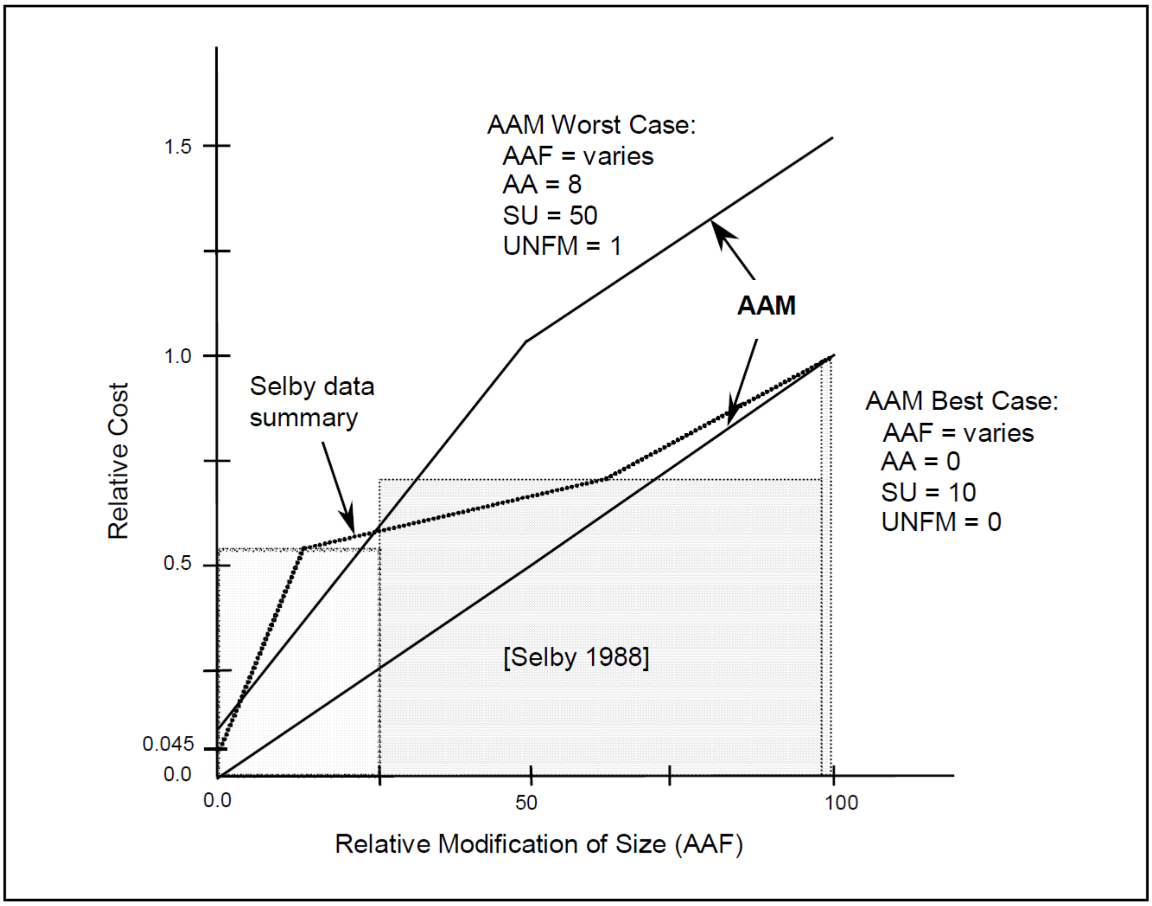
$$AAF = (0.4 \times DM) + (0.3 \times CM) + (0.3 \times IM)$$

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

Automated translation

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

AAM Range



See example at p. 57

Backfiring

How to move from the FP to LOC

Linguaggio	Livello nominale	LOC per Function Point		
		Minimo	Media	Massimo
1st Generation	1.00	220	320	500
Basic assembly	1.00	200	320	450
Macro assembly	1.50	130	213	300
C	2.50	60	128	170
BASIC (interpreted)	2.50	70	128	165

See 2nd and 3rd generation and for DB or Smalltalk

LOC estimation

See example at p. 65

UFP/LOC

Table 4. UFP to SLOC Conversion Ratios

Language	Default SLOC / UFP	Language	Default SLOC / UFP
Access	38	Jovial	107
Ada 83	71	Lisp	64
Ada 95	49	Machine Code	640
AI Shell	49	Modula 2	80
APL	32	Pascal	91
Assembly - Basic	320	PERL	27
Assembly - Macro	213	PowerBuilder	16
Basic - ANSI	64	Prolog	64
Basic - Compiled	91	Query – Default	13
Basic - Visual	32	Report Generator	80
C	128	Second Generation Language	107
C++	55	Simulation – Default	46
Cobol (ANSI 85)	91	Spreadsheet	6
Database – Default	40	Third Generation Language	80
Fifth Generation Language	4	Unix Shell Scripts	107
First Generation Language	320	USR_1	1
Forth	64	USR_2	1
Fortran 77	107	USR_3	1
Fortran 95	71	USR_4	1
Fourth Generation Language	20	USR_5	1
High Level Language	64	Visual Basic 5.0	29
HTML 3.0	15	Visual C++	34
Java	53		

Language	QSM SLOC/FP Data			
	Avg	Median	Low	High
ABAP (SAP) *	28	18	16	60
ASP*	51	54	15	69
Assembler *	119	98	25	320
Brio +	14	14	13	16
C *	97	99	39	333
C++ *	50	53	25	80
C# *	54	59	29	70
COBOL *	61	55	23	297

Exercise

See in SE_11bis_COCOMOex.pdf