Cost Models

Cost models are used to estimate manpower and schedule based on the software project profile. Three major categories are used for estimation techniques in the computer models:

- Regression-based models are derived and analyzed from historical data to express mathematical relationships among project variables. Examples are Barry Boehm's COCOMO and Ray Kile's REVIC.
- Empirical-based models are derived from observing the past performance of projects' from similar domains. Capers Jones' Checkpoint is an example.
- Theory-based models are based on underlying theoretical considerations for software development processes. These models are the most expensive as they require extensive research. Examples are Larry Putnam's SLIM and Lockheed Martin's PRICE-S.

Typical inputs to the models include software size attributes, product attributes, computer attributes, personnel attributes, and project attributes. Typical outputs from the models are labor cost in total staff months (effort), development time in accounting months (schedule), and the cost and schedule by activity. This chart shows a comparison of four cost models and the parameters required for that tool.

Model Parameters	SLIM	Price-S	COCOMO	Jensen
Size Attributes				
Source Instructions	✔		✔	✔
Number of routines		✔		
Number of data items			✔	
Documentation				✔
Number of personnel	✔			✔
Product Attributes				
Type	✔	✔		
Complexity	✔	✔	✔	✔
Language	✔			
Reuse	✔	✔	✔	✔
Required reliability	✔	✔	✔	✔
Computer Attributes				
Time constraint	✔	✔	✔	✔
Storage constraint	✔	✔	✔	✔
Hardware configuration		✔		
Development		✔	✔	✔
Personnel Attributes				

Model Parameters	SLIM	Price-S	COCOMO	Jensen
Personnel capability	✔	✔	✔	✔
Personnel continuity				
Hardware experience	✔	✔	✔	✔
Applications experience	✔	✔	✔	✔
Language experience	✔	✔	✔	✔
Project Attributes				
Tools and techniques	✔	✔	✔	✔
Requirements definition				✔
Requirements volatility		✔	✔	✔
Schedule	✔	✔	✔	✔
Security				
Computer access	✔		✔	✔
Travel/rehosting multisite		✔		✔
Support software maturity			✔	
Total	16	18	17	20

You will certainly note that each tool requires different parameters. You should also notice that no tool uses personnel continuity or security. Perhaps as future tools are created, these variables will be considered.

COnstructive COst MOdel

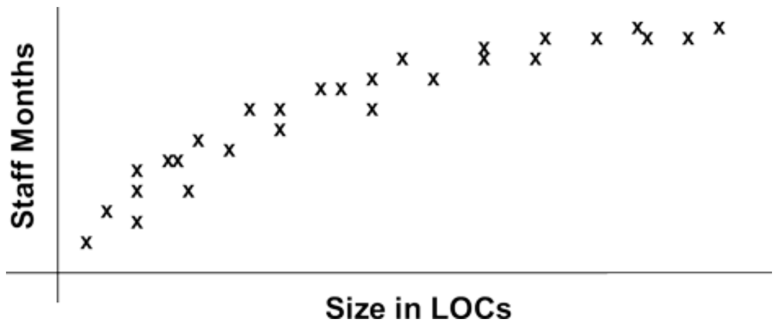
The **Co**nstructive **Co**st **Mo**del or COCOMO as it is generally referred to, is the most widely used software cost and schedule estimation tool still today. Barry Boehm, when he was working at TRW, developed COCOMO in the mid-1970s. It is non-proprietary; various versions can be found on the internet. As with all models, it is critical to know what activities are covered by COCOMO; it addresses:

- Software Design
- Code and Unit Test
- Unit and Computer Software Component (CSC) Integration into the larger Computer Software Configuration Items (CSCI)
- CSCI Test (or System Test) and Sell-off

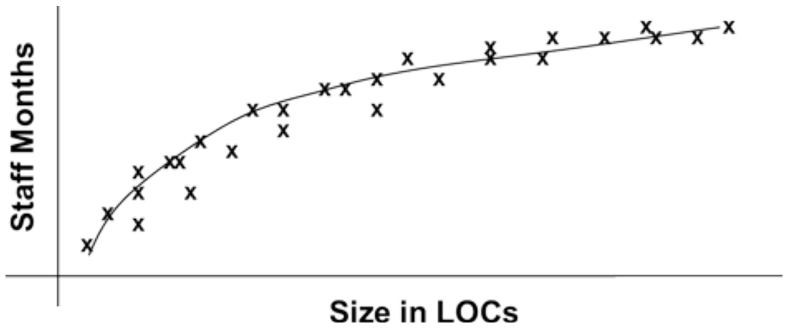
Note that software requirements analysis is not covered as COCOMO assumes that this activity is a system engineering function.

In 1987, Ray Kile updated COCOMO and developed the **Rev**ised **I**ntermediate **C**OCOMO (REVIC) to support Ada, software reuse, and incremental development model. Corporate experience has shown that the REVIC equations result in lower staffing levels and longer schedules. The original COCOMO and the even later revised COCOMO II are closer to company's experience. Some organizations have checked results with a Function Point model or other methods. For this reason, we will use COCOMO in its original form in our examples.

Barry Boehm collected data from 63 completed projects. He counted the LOC and obtained labor hours from the TRW time accounting system. He assumed 160 work hours in a month.



Then he performed Regression Modeling analysis to fit a curve to the data collected for 63 projects producing equations with coefficients and exponent to predict future staff months (or effort) based on estimated size.



To use the COCOMO tool, it requires the user to select a development mode for the project and one of the available models.

There are three development modes: Organic, Semi-detached, and Embedded:

1. Organic uses a small team in a familiar environment and with a familiar application. Generally the team is working to a set of less-than-rigid requirements, e.g., data reduction, scientific models, business models, simple inventory control, or simple production control.
2. Semi-detached is the intermediate mode. It has a mixture of organic and embedded mode characteristics; for example, most transaction processing systems, ambitious inventory system, ambitious inventory control, and ambitious production control would be semi-detached.
3. Embedded has tight constraints on software. There are complex interactions with hardware and the operating environment, the cost of validation and change is high.

Embedded mode includes unique applications, e.g., avionics, large and complex transaction processing, aircraft on-board collision avoidance system, and complex command and control systems.

The user also chooses one of three available models: Basic, Intermediate, and Detailed:

4. The Basic model uses a single equation for selected mode.
5. The Intermediate introduces multipliers (or project characteristics) which are incorporated into the cost driver equations.
6. The Detailed adds the phases of the project and subsystems to use the multipliers in the cost driver equations for each phase of project and/or for each subsystem.

The equations are:

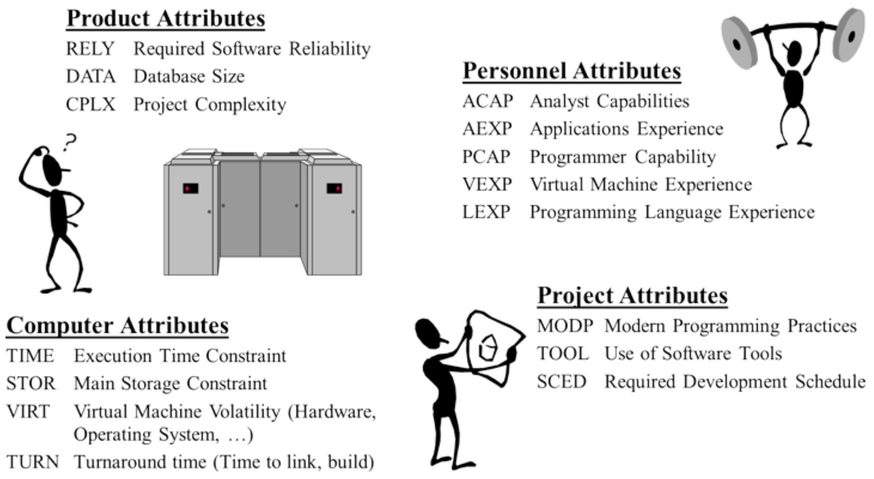
Model	**Mode**	**Effort**	**Schedule**
Basic	Organic	Semi-detached	Embedded
	$SM = 2.4 (KDSI)^{1.05}$	$SM = 3.0 (KDSI)^{1.12}$	$SM = 3.6 (KDSI)^{1.20}$
	$T_{DEV} = 2.5 (SM)^{0.38}$	$T_{DEV} = 2.5 (SM)^{0.35}$	$T_{DEV} = 2.5 (SM)^{0.32}$
Intermediate & Detailed	Organic	Semi-detached	Embedded
	$SM = 3.2 (PEM) (KDSI)^{1.05}$	$SM = 3.0 (PEM) (KDSI)^{1.12}$	$SM = 2.8 (PEM) (KDSI)^{1.20}$
	$T_{DEV} = 2.5 (SM)^{0.38}$	$T_{DEV} = 2.5 (SM)^{0.35}$	$T_{DEV} = 2.5 (SM)^{0.32}$

SM: Staff Months
PEM: Product of Effort Multipliers (covered next)
KDSI: Thousand Delivered Source Instructions
T_{DEV}: Time for Development

Note that the same equations are used for both the Intermediate and Detailed models. For the Detailed model, you would use these equations for each phase or each major subsystem. The user then runs the model, assesses the results, refines the parameters, and runs again as necessary.

Effort Multipliers

The effort multipliers are divided among four attribute categories: product, computer, personnel, and project.



Product Attributes

RELY	Required Software Reliability
DATA	Database Size
CPLX	Project Complexity

Personnel Attributes

ACAP	Analyst Capabilities
AEXP	Applications Experience
PCAP	Programmer Capability
VEXP	Virtual Machine Experience
LEXP	Programming Language Experience

Computer Attributes

TIME	Execution Time Constraint
STOR	Main Storage Constraint
VIRT	Virtual Machine Volatility (Hardware, Operating System, …)
TURN	Turnaround time (Time to link, build)

Project Attributes

MODP	Modern Programming Practices
TOOL	Use of Software Tools
SCED	Required Development Schedule

For the target machine, the user determines for each cost driver (or cost attribute) the effort multiplier rating that will be used and justifies the reason for choosing that rating. This table shows the various effort multiplier ratings:

Cost Driver	Very Low	Low	Nominal	High	Very High	Extra High
RELY	Slight inconvenience	Easily recover losses	Recoverable losses	High financial loss	Risk to human life	
DATA		D/P < 10	10 < D/P < 100	100 < D/P < 1000	D/P > 1000	
CPLX	Refer to next table					
TIME			< 50% use of avail exec time	70%	85%	95%
STOR			< 50% use of avail exec time	70%	85%	95%
VIRT		Major chg 1 per 12 mos, min; 1 per mo	Maj: 1/6 mos Min: 1/2 wks	Maj: 1/2 mos Min: 1/week	Maj: 1/2 wks Min: 1/2 days	
TURN		Interactive	< 4 hrs	4–12 hrs	> 12 hrs	
ACAP	15th percentile	25th percentile	55th percentile	75th percentile	90th percentile	
AEXP	< 4 mos experience	1 year	3 years	6 years	12 years	
PCAP	15th percentile	25th percentile	55th percentile	75th percentile	90th percentile	
VEXP	< 1 month	4 months	1 year	3 years		

Cost Driver	Very Low	Low	Nominal	High	Very High	Extra High
VEXP	< 1 month	4 months	1 year	3 years		
MODP	No use	Beginning use	Some use	General use	Routine use	
TOOL	Basic use of microprocessor tools	Basic use of small tools	Strong use of small tools and basic use of large tools	Strong use of large tools and use of test tools	Advanced use of large tools	
SCED	75% of nominal	85% of nominal	100% of nominal	130% of nominal	160% of nominal	

Note: for the Data row: D: Database size in bytes, P: LOC delivered

You will see the ratings go from Very Low to Extra High. Not all cost drivers used every rating. This reason is based on Barry Boehm's regression modeling analysis. These terms and their ratings are:

Required Software Reliability (RELY) is defined in terms of the effect it has on the user from a Slight Inconvenience (Very Low rating) to Risk to Human Life (Very High rating). As you can see there is no Extra High rating.

Database size (DATA) is the measure of data in the system, that is, the ratio D/P where D is the database size in bytes and P is the estimated delivered program instructions or delivered LOC. The rating is from Low or a ratio of less than 10 to Very High with a rating greater than 1000.

Product Complexity (CPLX) is comprised of four types of operations: Control Operations, Computational Operations, Device-Dependent Operations, and Data Management Operations. It is determined for each subsystem when using the Detailed Model. This cost driver spans the entire range from Very Low to Extra High. Here is a selection of the operations:

CPLX - Product Complexity (or each subsystem for Detailed Model)

Rating	Control Operations	Computational Operations	Device-dependent Operations	Data Management Operations
Very Low	Straight-line code with few non-nested SP operators: DOs, CASEs; Simple predicates	Evaluation of simple expressions A = B + C*D	Simple read/write statements with simple formats	Simple arrays in main memory
Low	Straightforward nesting of SP operators, mostly simple predicates	Evaluation of some complex expressions	Device timing-dependent coding	
Nominal	Mostly simple nesting, some intermodule control, decision tables	Structured numerical analysis		
High	Highly nested SP operators, compound predicates, queue & stack control	Unstructured evaluation of		
Very High	Reentrant & recursive coding, fixed priority interrupt handling			
Extra High	Multiple resource scheduling, dynamically changing priorities, microcode level control			

The Available Execution Time Constraint (TIME) is what percentage of the available execution time your system will use. It can range from less than 50% use of the available execution time (Nominal rating) to 95% (Extra High).

Similarly the Available Storage Constraint (STOR) is what percentage of the available storage your system will use. It can range from less than 50% use of the available storage (Nominal rating) to 95% (Extra High).

Virtual System Volatility (VIRT) is the anticipated number or major and minor changes to the system over a period of time. It ranges from a single major change every 12 months and one minor change each month for a Low rating to a major change every two weeks and a minor change every two days for a Very High rating. For a given software product, the underlying virtual machine is the complexity of the hardware and software (including the operating system and database management system it calls upon) to accomplish the tasks.

Computer Turnaround Time (TURN) is the time it takes to get a response back to the user. Most applications today are interactive. But there are applications which take a long time such as payroll systems which typically run overnight. This cost driver ranges from Interactive at Low to greater than 12 hours at Very High. Since COCOMO can be calibrated to your company's operating norms, this is probably a cost driver that would change.

Analyst Capability (ACAP) is the average percentile of the analysts' capability working on this project ranging from 15% or Very Low to 90% or Very High.

Application Experience (AEXP) is the average level of experience working with in this application domain. It ranges from a team with less than four months experience (Very Low) to a team with 12 years' experience (Very High.)

Programmer Capability (PCAP) is the average percentile of the programmer or software developers' capability working on this project ranging from 15% or Very Low to 90% or Very High.

Virtual Machine Experience (VEXP) is the team's experience with virtual machines. This is another parameter that is less useful in today's environment. It ranges from less than one month (Very Low) to three years (High.)

Program Language Experience (LEXP) is the team's experience with the specific programming language. It ranges from less than one month (Very Low) to three years (High.)

Modern Programming Practices (MODP) are the level of use of current programming practices. It ranges from No Use (Very Low) to Routine Use (Very High.)

The Use of Tools or Tools Experience (TOOL) is the level and number of tools available to the team. It ranges from basic use of microprocessor tools (Very Low), basic use of small tools (Low), strong use of small tools and basic use of large tools (Nominal), strong use of large tools and use of test tools (High) to advanced use of large tools (Very High.)

Lastly the Required Development Schedule (SCED) is the schedule that the customer requires. Ideally the team should deliver the product in the nominal time frame or at the exact time due. If the product is delivered in 75% of the time, it is rated Very Low. If it is delivered late at 160% of nominal, it is rated (Very High.)

The Effort Multiplier ratings are overlaid on the chart to reveal these Effort Multiplier Values.

Attributes		Very Low	Low	Nominal	High	Very High	Extra High
Product Attributes							
RELY	Required Software Reliability	.75	.88	1.00	1.15	1.40	
DATA	Database Size		.94	1.00	1.08	1.16	
CPLX	Product Complexity	.70	.85	1.00	1.15	1.30	1.65
Computer Attributes							
TIME	Execution Time Constraint			1.00	1.11	1.30	1.66
STOR	Main Storage Constraint			1.00	1.06	1.21	1.56
VIRT	virtual Machine		.87	1.00	1.15	1.30	

Attributes		Very Low	Low	Nominal	High	Very High	Extra High
	Volatility						
TURN	Computer Turnaround Time		.87	1.00	1.07	1.15	
Personnel Attributes							
ACAP	Analyst Capability	1.46	1.19	1.00	.86	.71	
AEXP	Applications Experience	1.29	1.13	1.00	.91	.82	
PCAP	Programmer Capability	1.42	1.17	1.00	.86	.70	
VEXP	Virtual Machine Experience	1.21	1.10	1.00	.90		
LEXP	Programming Language Experience	1.14	1.07	1.00	.95		
Project Attributes							
MODP	Use of Modern Programming Practice	1.24	1.10	1.00	.91	.82	
TOOL	Use of Software Tools	1.24	1.10	1.00	.91	.83	
SCED	Required Development Schedule	1.23	1.08	1.00	1.04	1.10	

SM: Staff Months
PEM: Product of Effort Multipliers (covered next)
KDSI: Thousand Delivered Source Instructions
T_{DEV}: Time for Development

Note that the same equations are used for both the Intermediate and Detailed models. For the Detailed model, you would use these equations for each phase or each major subsystem. The user then runs the model, assesses the results, refines the parameters, and runs again as necessary.

Cost Trade-Off Analysis

COCOMO can be used to compare alternative approaches and the cost and schedule impacts. Using the previous example, we can determine if the use of average analysts and programmers each earning a salary of $5K/SM is more cost effective than using more experienced analysts and programmers with a salary of $6K/SM.

Trade Data	More Experienced Personnel	Average Personnel

Trade Data	**More Experienced Personnel**	**Average Personnel**
Cost/Staff Month	$6K/Staff Month	$5K/Staff Month
Analyst Capability	0.86	1.0
Programmer Capability	0.86	1.0
Effort Adjustment Factor	1.17	1.17 / (0.86 * 0.86) = 1.58
Estimated Effort	52 Staff Months	70 Staff Months
Estimated Cost	$6K/SM * 52 SM = $312K	$5K/SM * 70 SM = $350K

1.17/52 = 1.58/X
X = (1.58)*(52)/(1.17)
X = 70