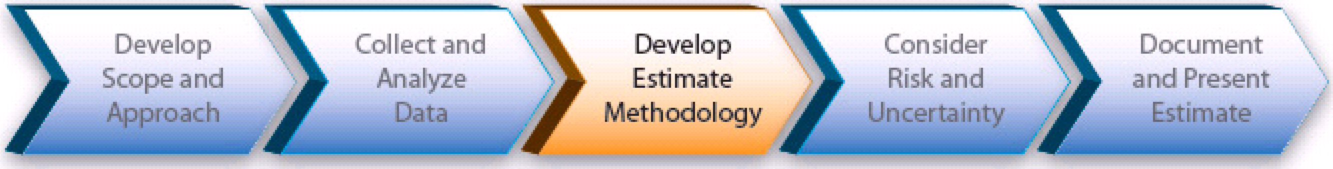
Introduction and Objectives

Welcome to the Develop Estimate Methodology lesson. After completing this lesson, you will understand the third of the five major steps of developing a software cost estimate.

Lesson Objectives

- Evaluate the use of parametric, analogy, and other techniques in software cost estimating.
- Summarize the use of cost estimating relationships (CERs) and schedule estimating relationships (SERs) in software estimating.
- Consider appropriate time phasing for a software estimate.
- Critique the use of off-the-shelf (OTS) models for software estimating.


Develop Scope and Approach
Collect and Analyze Data
Develop Estimate Methodology
Consider Risk and Uncertainty
Document and Present Estimate

D

Long Description

Graphic illustrates the steps of the Cost Estimating process. The steps from left to right are:
Develop Scope and Approach, Collect and Analyze Data, Develop Estimate Methodology (highlighted), Consider Risk and Uncertainty, and Document and Present Estimate.

Parametric vs. Analogy

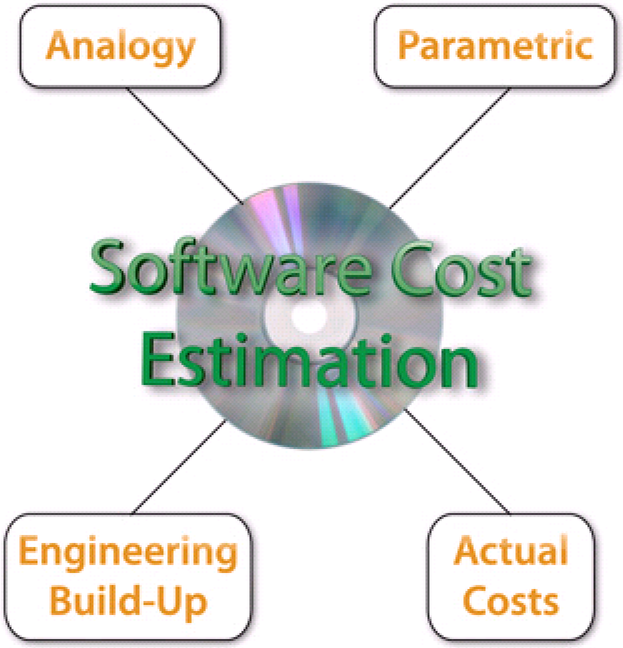
The first consideration in estimating software cost and effort (labor hours or person-months) is to choose the estimation method.

There are four major analytical methods or cost estimating techniques:

- Analogy
- Parametric
- Engineering Build-Up
- Actual Costs

Of these, the first two are most important for software cost estimating. Parametric is strongly preferred where the data exist to support it, but Analogy is a perfectly acceptable fallback.

Expert Opinion, which is not listed, is not admitted as a stand-alone technique, though certainly subject matter experts (SMEs) can help to identify and interpret historical program data and advise on the application of methodologies.



Questions Managers Should Ask

NOTE

D

Popup Text

Questions Managers Should Ask

Are the estimated costs and schedule consistent with demonstrated accomplishments on other projects?

- Did the estimate follow the organization's structured and documented process for relating estimates to actual costs and schedules of completed work?
- Did cost and schedule estimates quantify demonstrated organizational performance in ways that normalize for differences among software products and projects (so that a simple, unnormalized, lines-of-code per staff-month extrapolation is not the basis for the estimate)?
- Did extrapolations from past projects account for differences in application technology and the effects of introduction of new software technology of processes?
- Did extrapolations from past projects account for observed, long-term trends in software technology improvement?

Have steps been taken to ensure the integrity of the estimating process?

- Did management review and agree to the values for all descriptive parameters before costs were estimated?
- Has more than one cost model or estimating approach been used, and were the differences in results analyzed and explained?
- Were people from related but different projects or disciplines involved in preparing the estimate?

Note

Keep in mind that different techniques can be used for different cost elements; that different techniques can be used throughout the program life cycle; and that a cross-check using a second technique can significantly strengthen an estimate.

Long Description

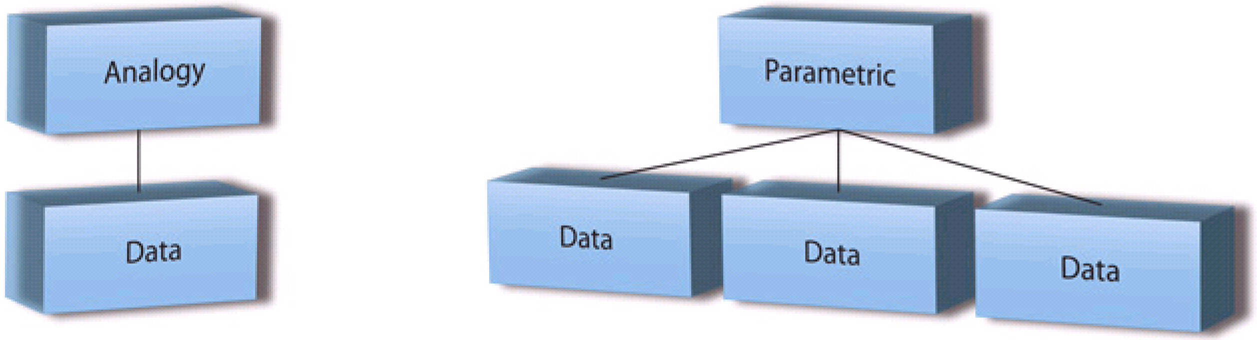
Software Cost Estimation the center of the graphic with Analogy, Parametric, Engineering Build-Up and Actual Costs linked to it with lines.

Parametric vs. Analogy, Cont.

It is a myth that analogy is the only alternative early in a program, when some aspects of the system to be acquired are not yet well-defined.

Analogy and parametric require the exact same inputs (and therefore definition for the program being estimated), but the difference is analogy only requires one historical data point (comparable program), whereas parametric requires many. Thus, the prevalence of analogy estimating early in development is more of a practical than a theoretical consideration.

Often, investment in historical data collection happens in parallel with estimate development, so that as the supporting database is built and scrubbed, parametric becomes a more viable option. In short, parametric is preferred because it offers measures of significance (confidence in using reasonable cost drivers) and uncertainty (the plus or minus around the estimate follows directly from the data set), both of which analogy lacks.



D

Long Description

Two flowcharts, one Analogy and the other Parametric. The Analogy flowchart has one subordinate Data branch and the Parametric flowchart has three subordinate Data branches.

Parametric vs. Analogy - Analogy Methodology

Analogy is the simplest form of estimating, capturing the essential thought process of all estimates. It draws a direct comparison between the program being estimated and a similar, completed program for which historical information is available. (Two or three comparable historical programs is a bit of a gray area between analogy and parametric, but in a classical analogy there is only a single comparable.)

The cost for the historical program is adjusted by a ratio of one or more parameters to capture the differences between the two. There parameters should be as objective as possible.

For example, a previous development project produced 10 KSLOC with 50 person-months (PM) of effort. If the current project is expected to generate 20 KSLOC, then the estimated effort would be 50 * (20/10) = 100 PM. In this case, the math is simple enough we can do it in our heads – twice the code is estimated to require twice the effort. This is illustrated in the graphic below.

Previous		Current	
Person Months (PM)	KSLOC	Person Months (PM)	KSLOC
50	10	(?)	20

(?)= 50 X (20/10) = 100 PM

Note that this is tantamount to applying a productivity rate (in this case 200 SLOC/PM), however, caution should be practiced as this approach ignores fixed costs and economies or diseconomies of scale.

The analogy method is typically performed early in the cost estimating process, and is also commonly used for cross-checking more detailed estimates (i.e., sanity check).

D

Popup Text

Analogy Cost Estimate

An estimate of costs based on historical data of a similar (analogous) item.

Long Description

Table with the following data:

Previous		Current	
Person Months (PM)	KSLOC	Person Months (PM)	KSLOC
50	10	(?)	20

(?) = 50 X (20/10) = 100 PM

Parametric vs. Analogy - Analogy Advantages

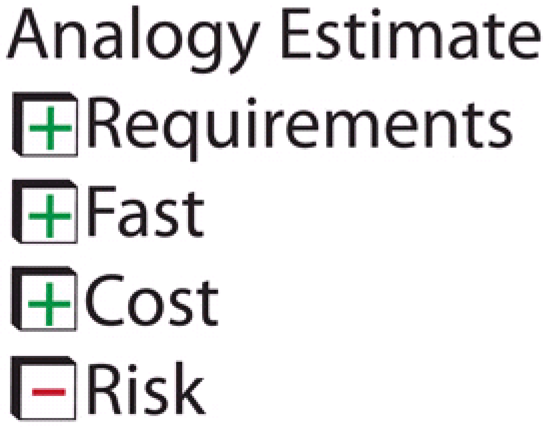
Estimating by analogy has many advantages.

With emerging technologies and the rapidly evolving environment of software/IT, the number of comparable historical programs may be limited, but analogy requires only one.

Analogy is reasonably fast and inexpensive, and easy to change.

However, an estimate produced by analogy typically includes a high degree of cost risk because it is based on a single historical data point (we know not whether "lucky" or "unlucky," except perhaps anecdotally) and tends to require subjective judgment in the selection of the comparable program and scaling quantity.

Estimating by parametrics is one way to address some of this cost risk.



D

Long Description

List of Analogy Estimate advantages which include:

- Requirements (Positive)
- Fast (Positive)
- Cost (Positive)
- Risk (Negative)

Parametric vs. Analogy - Parametric Methodology

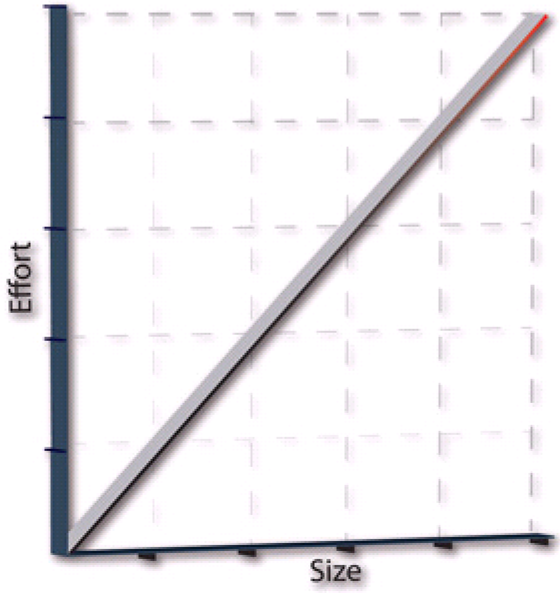
The Parametric technique is the most commonly used and most important for software estimation.

Sometimes called "statistical," parametric applies regression analysis to a database of several similar systems to develop cost estimating relationships (CERs).

CERs are equations that estimate the value of the dependent variable (cost or effort) based on the values of independent variables, or input parameters, such as software size.

Prime candidates for these parameters are those program characteristics that capture the cost drivers (Size, Complexity, Capability).

When a software CER is applied, it is important to know which activities are included in the predicted effort and which must be estimated separately.



Popup Text

Parametric Cost Estimate

A cost estimating methodology using statistical relationships between historical costs and other program variables such as system physical or performance characteristics, contractor output measures, or manpower loading.

Parametric vs. Analogy - Parametric Advantages

The parametric method has many advantages over other estimating methods.

Because CERs are based upon more than a single data point, estimating by parametrics is less risky than estimating by analogy, and it also quantifies the uncertainty in the CER, which can then be fed into the cost risk analysis.

Statistical significance allows us to be confident in the cost-driving parameters we are using.

The primary challenge in implementing the parametric approach is creating and maintaining the supporting normalized database of historical programs.

Parametric Estimate

+ Reduced Risk
+ Quantified Uncertainty
− Normalization

D

Long Description

List of Parametric Estimate advantages which include:

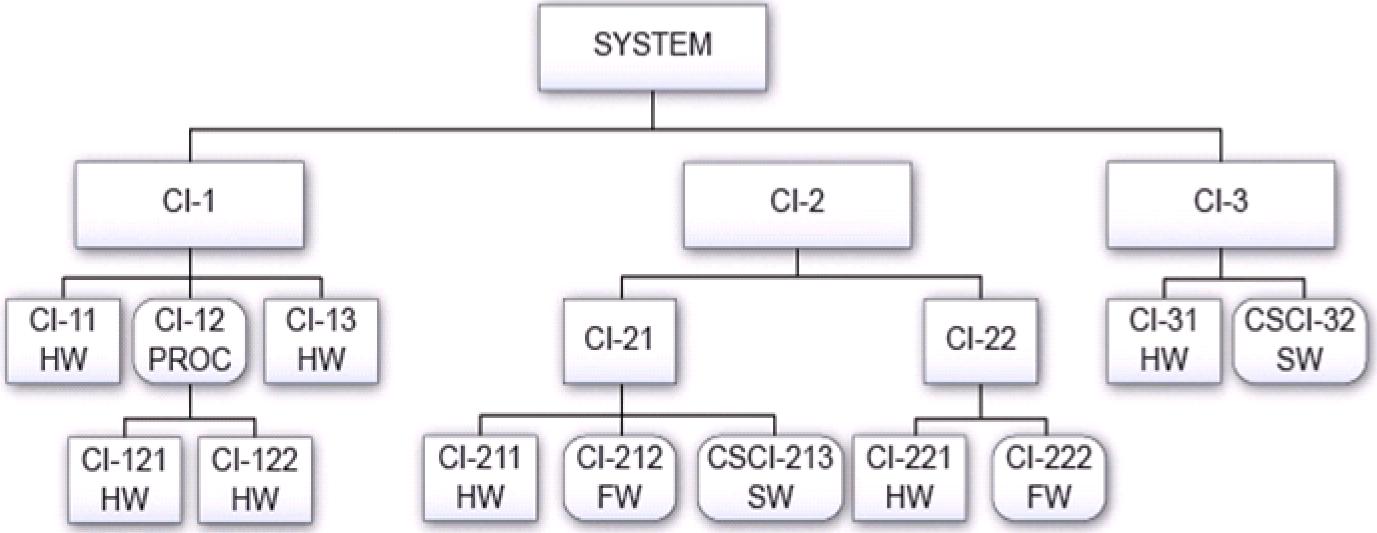
- Reduced Risk (Positive)
- Quantified Uncertainty (Positive)
- Normalization (Negative)

Parametric vs. Analogy - Application

Whether applying the analogy or the parametric cost estimating technique, it is important to properly structure and test the estimate.

Using the Work Breakdown Structure (WBS) to decompose the system and applying the appropriate costing techniques, the estimator should begin developing estimates for each of the cost elements. Keep in mind that a software estimate is usually part of a larger estimate.

"Testing" the estimate may include cross-checking the results with historical data from similar programs/systems; applying a different estimating methodology; and applying a different cost model. The estimate can also be compared to industry rules of thumb or benchmarks.

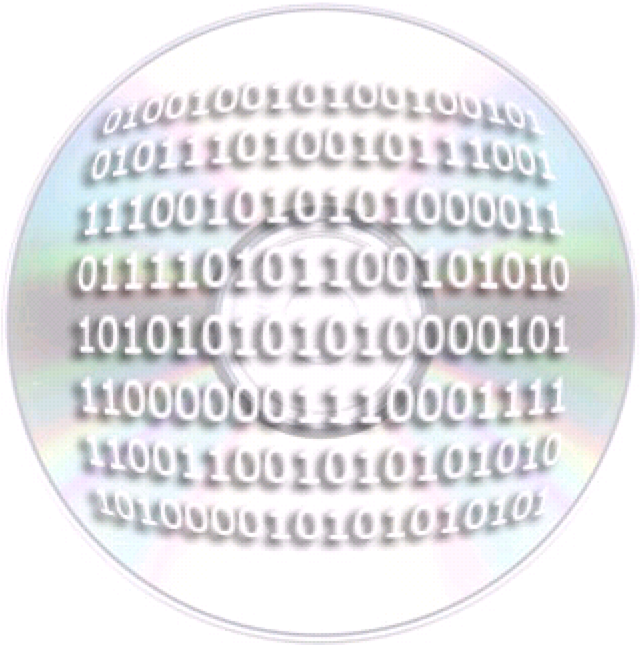


Other Methodologies - Engineering Build- Up

Because software is intangible, Engineering Build-Up is not really applicable in the same way it is implemented for hardware in a manufacturing environment.

Perhaps the closest equivalent is Function Point counting, which requires a detailed design similar to other build-up methods, but there are no corresponding labor standards for "building" the code to implement each function point.

Similarly, there is some affinity with productivity-based methods, though there would have to be a decomposition to the CSU level with specific productivities at that level for the comparison not to be strained.



Popup Text

Engineering Build-Up

Derived by summing detailed cost estimates of the individual work packages and adding appropriate burdens. Usually determined by a contractor’s industrial engineers, price analysts, and cost accountants.

Other Methodologies - Actual Costs

The Actual Costs technique extrapolates future estimated costs from actual costs, similar to analogy but based on data from the same program.

It is often called "Extrapolation" or "Extrapolation from Actuals." There are three variants: average, learning curve and estimate at completion. Select each tab below to read more.

Learning Curve | Estimate At Completion

One variant is just to use an average. In the ProRad example, if all waveforms were created equal, and the first ten were complete, we could simply estimate the per-waveform cost of the remaining 21 as an average of those first ten.

Popup Text

Extrapolation from Actual Costs

Extrapolation method requires prototype or preproduction actual cost data on the system considered. Primarily used in estimating the production cost of system hardware, and assumes a relationship (technical, performance) between cost of prototypes and production units.

Average

One variant is just to use an average. In the ProRad example, if all waveforms were created equal, and the first ten were complete, we could simply estimate the per-waveform cost of the remaining 21 as an average of those first ten.

Learning Curve

The second variant is learning curve, which applies only to recurring production and therefore is not germane to software.

Estimate At Completion

The third, and most applicable, is the Estimate At Completion (EAC) generated when conducting earned value management (EVM) on an in-process development effort.

Estimate At Completion (EAC)

Actual direct costs, plus indirect costs or costs allocable to the contract, plus the estimate of costs (direct and indirect) for authorized work remaining.

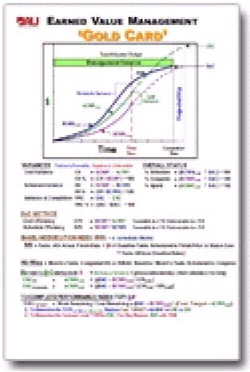
Other Methodologies - Actuals Cost, Cont.

Actuals is generally considered a high-fidelity technique for more concrete items such as hardware, since it uses the latest data from the program itself. Its application to software is a challenge because it is intangible, process-intensive, and there are often no definable end items until the development is complete. This is a more acute case and of the primary concern with EVM EACs in general, that performance on work to date may not be indicative of performance on the remaining work.

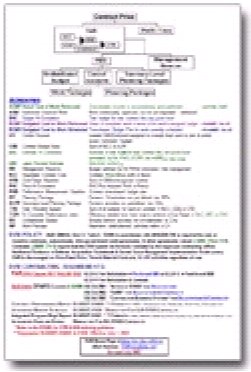
For example, if the development team performed only a cursory design effort and "declared victory," their earned value metrics may look quite good heading into coding, but problems may arise later in testing.

This concern is mitigated if the organization has an established track record of how EACs change over time throughout similar development efforts. If the Actuals method is used, the application of statistical analysis beyond the standard DAU Gold Card formulae is encouraged.If successful, the same benefits can result as in traditional regression-based CERs.

Finally, since Actuals requires the development effort to be significantly underway, it cannot be used beforehand for determining budgets. Select the images below to view the DAU Gold Card.



DAU Earned Value Management 'Gold Card'



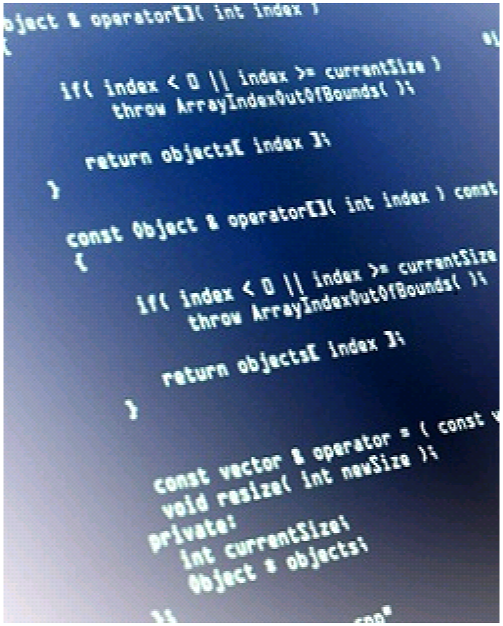
Cost Estimating Relationships (CERs)

The primary CERs of interest are those that estimate the "core" software development activities, ideally requirements through software test. As previously noted, this requires a historical database of software programs where the effort has been normalized to the same set of activities, and size expressed in a common measure such as ESLOC. Other factors that affect effort may be handled in one of several ways:

Additional Variables | Segregation | Indicator Variables

If different CSCIs were coded in different languages, for example, the ESLOC of one could be adjusted so that it reflects an equivalent effort in the other language.

Note that such a conversion factor may be difficult to determine empirically.


```
bject & operator[]( int index )
{
    if( index < 0 || index >= currentSize )
        throw ArrayIndexOutOfBounds( );

    return objects[ index ];
}

const Object & operator[]( int index ) const
{
    if( index < 0 || index >= currentSize
        throw ArrayIndexOutOfBounds( );

    return objects[ index ];
}

const vector & operator = ( const v
void resize( int newSize );
private:
    int currentSize;
    Object * objects;
```

It is generally preferred to develop your own CERs, but CERs validated by industry sources or funded by the government, such as the service cost centers or federally-funded research and development corporations (FFRDCs) like Rand and MITRE are also available.

Popup Text

Cost Estimating Relationship (CER)

A mathematical relationship that defines cost as a function of one or more parameters such as performance, operating characteristics, physical characteristics, etc.

Normalization

If different CSCIs were coded in different languages, for example, the ESLOC of one could be adjusted so that it reflects an equivalent effort in the other language.

Note that such a conversion factor may be difficult to determine empirically.

Additional Variables

Additional explanatory variables usually referred to as independent variables, can be introduced, such as average years of experience of the development team.

It is desired that such variables prove to be statistically significant.

Segregation

Data sets may be separated and distinct CERs run on each.

For example, if Ground Systems and Aircraft have fundamentally different software productivities, each could have its own CER.

In this case, the analyst is trading off more data points (degrees of freedom) and gaining (presumably) “tighter” sets in return.

Popup Text

Indicator Variables

In this “have-your-cake-and-eat-it-too” approach, the data set remains undivided, maintaining the advantage of a high number of degrees of freedom, but an indicator variable is added to account for the difference in the two populations, either as an adder (for additive CERs) or a factor (for multiplicative CERs).

This is generally preferable to segregation, though if the two populations are not statistically distinct enough, a simple combined CER with no indicator variable may prove to be superior.

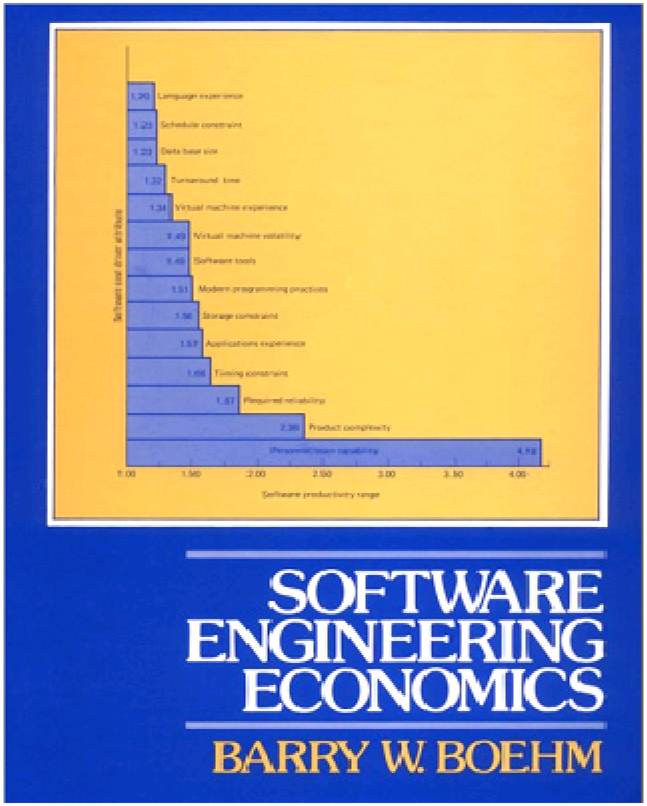
Cost Estimating Relationships (CERs) - COCOMO II CER

The COnstructive COst MOdel (COCOMO®) is used for estimating cost, effort, and schedule when planning a new software development activity.

COCOMO is an academic (not commercial), publicly-available model, and its CERs are open for inspection, avoiding the "black box" objection to commercial models.

The original COCOMO model was first published in 1981 and subsequently updated as COCOMO II to reflect changes in the software development process, including:

- Increases in desktop processing
- Code reuse
- Management of the software development effort


1.20 Language experience
1.23 Schedule constraint
1.23 Data base size
1.32 Turnaround time
1.34 Virtual machine experience
1.49 Virtual machine volatility
1.49 Software tools
1.51 Modern programming practices
1.56 Storage constraint
1.57 Applications experience
1.66 Timing constraint
1.87 Required reliability
2.36 Product complexity
Personnel/team capability 4.18
1.00 1.50 2.00 2.50 3.00 3.50 4.00
Software productivity range
Software cost driver attribute
SOFTWARE ENGINEERING ECONOMICS
BARRY W. BOEHM

Popup Text

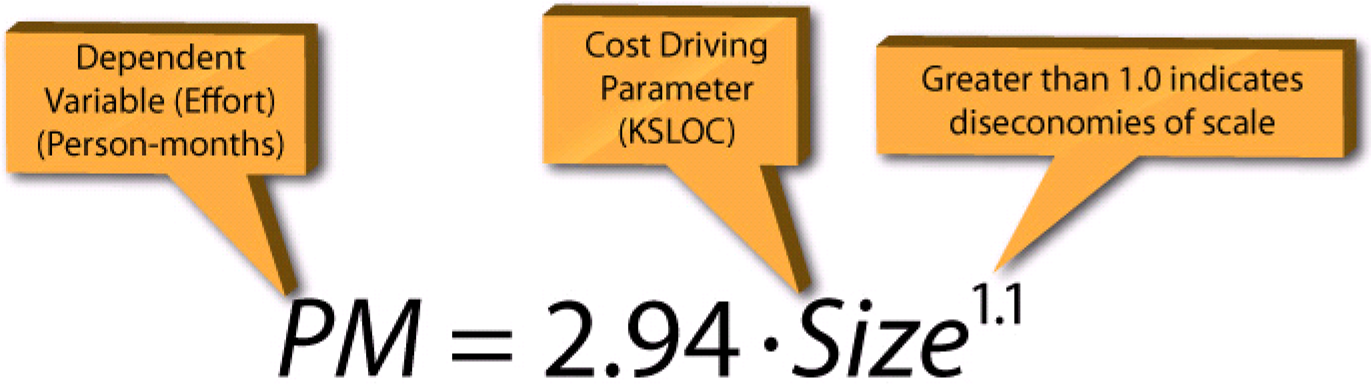
COnstructive COst MOdel (COCOMO®)

The second generation of estimating models put out by Barry Boehm and others based on their work at the University of Southern California (USC).

Cost Estimating Relationships (CERs) - COCOMO II CER, Cont.

The graphic illustrates a simplified version of the primary COCOMO II CER, with factors accounting for Complexity and Capability set to nominal values. There is a single cost-driving parameter, Size, which is expressed in KSLOC.

The exponent of 1.1, being slightly greater than one, indicates modest diseconomies of scale, so that effort increases at an increasing rate as size growth. The dependent variable is effort, expressed in person-months (PM).


Dependent Variable (Effort) (Person-months)
Cost Driving Parameter (KSLOC)
Greater than 1.0 indicates diseconomies of scale

$$PM = 2.94 \cdot Size^{1.1}$$

D

Long Description

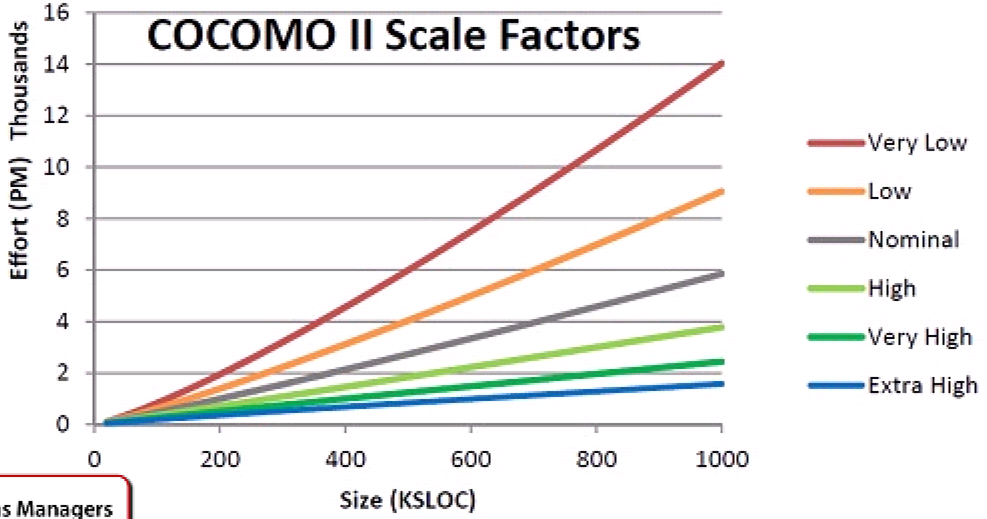
Labeled graphic of primary COCOMO II CER. The primary CER is PM = 2.94 * Size to the 1.1. PM labeled Dependent Variable (Effort) (Person-months); Size is labeled Cost Driving Parameter (KSLOC) and exponent 1.1 is labeled Greater than 1.0 indicates diseconomies of scale.

Cost Estimating Relationships (CERs) - Response to Size

This graph shows the behavior of cost (effort) as a function of size. As expected, effort increases as size increases. In many cases, the response is approximately linear, but this can change markedly depending on factors such as how well the project is managed.

As previously described, when Diseconomies of Scale are present, the exponent on size is greater than one, and the graph curves upward (second derivative positive!), whereas when Economies of Scale are present, the exponent is less than one and the graph curves downward (second derivative negative).

Again, experience has shown that most projects are prone to diseconomies of scale, so be wary of any software estimate that does not include them.



Questions Managers Should Ask

Popup Text

Questions Managers Should Ask

Has the task been appropriately sized?

- Have structured and documented processes been used to estimate and describe the size of the software product, and to estimate and describe the extent of reuse?
- Is the sizing estimate based on a solid understanding of both defined and emerging requirements?
- Have the descriptions of size and reuse identified what is included in (and excluded from) the size and reuse measures used?
- Do the measures of reuse distinguish between code that will be modified and code that will be integrated as-is into the system?
- Are the definitions, measures, and rules used to describe size and reuse consistent with the requirements (and calibrations) of any models used to estimate cost and schedule?
- Was the size estimate checked by relating it to measured sizes of other software products or components?
- Was the size estimating process checked by testing its predictive capabilities against measured sizes of completed products?

Are the estimated costs and schedule consistent with demonstrated accomplishments on other projects?

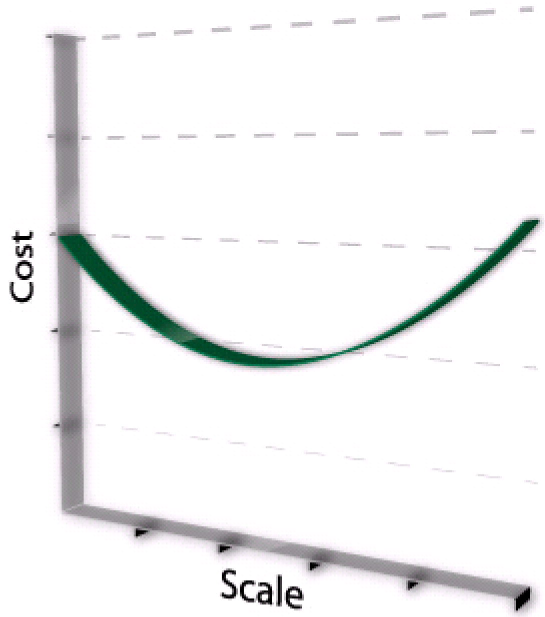
- Did the methods used to account for reuse recognize that reuse is not free (so that the estimate accounts for activities such as interface design, modification, integration, testing, and documentation that are associated with effective reuse)?

Cost Estimating Relationships (CERs) - Diseconomies of Scale

Diseconomies of scale can be mitigated somewhat by the use of software tools and a collocated collaborative environment, captured in the previous Capability discussion as the TOOL (Use of Software Tools) and SITE (Multisite Development) factors in COCOMO II.

However, since these are applied as simple multipliers in the estimate, they do not change the exponent, so that while high ratings can reduce the estimate for a give size, the response of the estimate to size will continue to reflect the diseconomies of scale as size grows.

The multipliers are discussed next.



Cost Estimating Relationships (CERs) - Response to Capability and Complexity

Not shown in the simplified COCOMO II CER are a number of effort multipliers (EMs), that can be used to scale costs linearly after the effect of size is captured. Though part of a Parametric equation (CER), these EMs reflect more of an Analogy thought process.

For example, if a software product rates High (instead of Nominal) on the Required Software Reliability (RELY) factor, a multiplier of 1.15 (instead of 1.0) is applied. What we are saying is that requiring higher reliability increases software development costs by 15%. The effect for Very High RELY is even more pronounced, with a 1.40 factor (or 40% adder).

The EMs can also be viewed as a sort of calibration step wherein the essential cost-driving relationship with Size is improved by taking into account these other factors. The ratings (Very Low / Low / Nominal / High / Very High / Extra High) that translate into EMs represent an ordinal scale (Nominal is "greater than" Low), but neither an interval nor a ratio scale. That is, the difference between High and Nominal is not equal to the difference between Nominal and Low, nor is High (4th on the scale) twice Low (2nd on the scale).

Questions Managers Should Ask

D

Popup Text

Questions Managers Should Ask

Are the estimated costs and schedule consistent with demonstrated accomplishments on other projects?

- Does the estimate reflect the actual capability and demonstrated productivity of the software development organization (or an appropriate range if the organization is not yet known)?

Long Description

Two arrows pointing upward, one 15% Cost labeled High (RELY) and the other 40% Cost labeled Very High (RELY).

Cost Estimating Relationships (CERs) - ProRad CER Example

The below table shows the simplified COCOMO II CER applied to the ProRad SLOC counts for key performance parameters (KPP) waveforms with all new code, with nominal parameter values. Estimates are in person-months (PM).

The cost column gives the equivalent at a notional fully-burdened labor rate of $16K/PM. The effort estimates may need to be adjusted up or down as the Complexity and Capability associated with each waveform and its developer, respectively, depart from the nominal values inherent in the simplified CER. Note that these costs are lower than those shown later on for these waveforms, developed using a different model.

Waveform	Req Type	Difficulty	KSLOC	New	Effort (PM)	Cost ($M)
EPLRS	KPP	3	75	No		
Wideband Digital Waveform	KPP	3	180	Yes	889.5	$14.2
SINCGARS ESIP	KPP	2	53	No		
UHF DAMA/DASA SATCOM (181, 182, 183)	KPP	2	88	Yes	404.8	$6.5
UHF DAMA/DASA SATCOM (184)	KPP	2	30	Yes	123.9	$2.0
HAVE QUICK I/II (UHF)	KPP	1	5	Yes	17.3	$0.3
Link 16	Threshold (T)	3	100	Yes	466.0	$7.5

Viewing these COCOMO results as a cross-check, you would want to ask questions as to why the primary estimates are significantly higher: Inclusion of additional activities? Higher factors for difficulty/complexity? Addition of code growth and other risk? Application of a higher labor rate? All of the above? It all comes back to the question asked in the data normalization section: "What's in the number?"

D

Long Description

Table with the following data:

Waveform	Req Type	Difficulty	KSLOC	New	Effort (PM)	Cost ($M)
EPLRS	KPP	3	75	No		
Wideband Digital Waverform	KPP	3	180	Yes	889.5	14.2
SINCGARS ESIP	KPP	2	53	No		
UHF DAMA/DASA SATCOM (181, 182, 183)	KPP	2	88	Yes	404.8	6.5
UHF DAMA/DASA SATCOM (184)	KPP	2	30	Yes	123.9	2.0
HAVE QUICK I/II (UHF)	KPP	1	5	Yes	17.3	.3
Link 16	Threshold (T)	3	100	Yes	466.0	7.5

Cost-on-Cost CERs

Some supporting activities such as Systems Engineering and Program Management (SE/PM), and their associated so-called "below-the-line" (BTL) costs, are often seen as being driven by the core effort – software development, in this case.

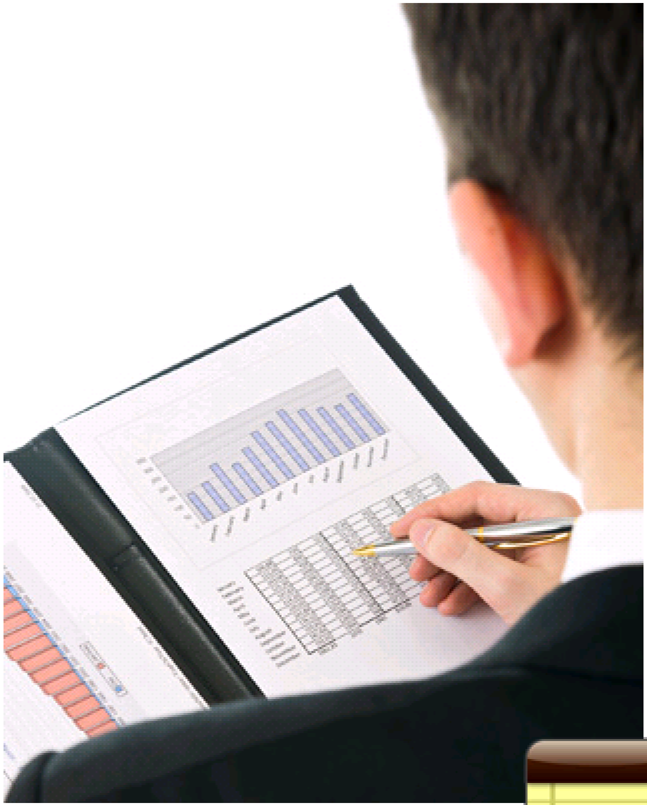
These may then be estimated by cost-on-cost CERs, meaning that both the input to and output from the CER are costs (or effort in hours).

In deriving the CER, the cost input values are historical actuals, but in applying the CER, the cost input must be estimated first by one of the previously-discussed methods.

Software-specific below-the-lines may include things like configuration management (CM) and independent verification and validation (IV&V).

IV&V is roughly the software equivalent of Quality Assurance (QA) in manufacturing, which is also often estimating as a below-the-line.

Depending on the program WBS, more general below-the-lines such SE/PM may be driven by software development together with hardware development costs, for example.



NOTE

Popup Text

Independent Verification and Validation (IV&V)

An independent review of software performed by an organization that is technically, managerially, and financially independent of the development organization.

Note

Note that a full-fledged CER based on a representative data set is preferred to a simple factor.

The former is allowed to have a non-zero y-intercept, and while we caution against strictly interpreting this as a fixed cost, the idea is that the additional degree of freedom allows the CER to better reflect the driving relationship without requiring a fixed percentage for all sizes of project.

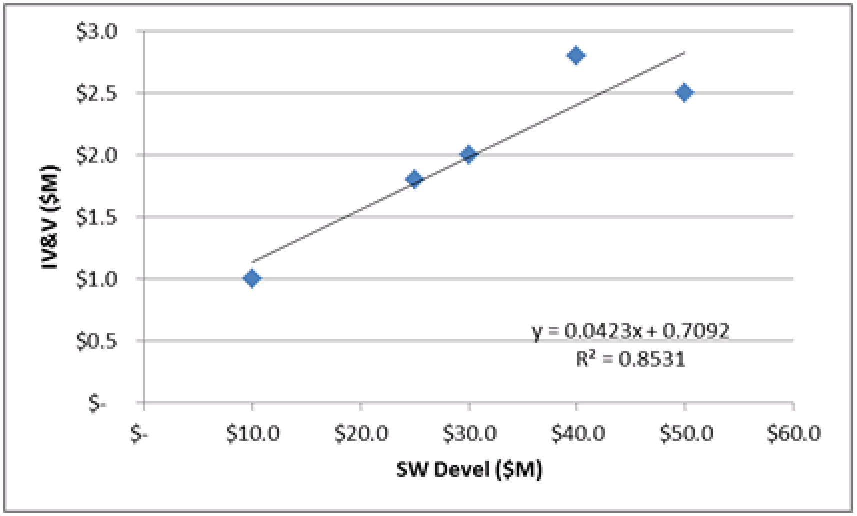
Cost-on-Cost CERs - ProRad SE/PM Example

The data table below left shows core software development together with IV&V activities for five historical programs deemed comparable to ProRad. In additional columns IV&V is expressed as a percentage of core software development effort for each of the individual programs.

Rather than take the average of these percentages (or worse yet "cherry-pick" one of them), a regression can be run to express each of the below-the-line (BTL) cost elements as a simple linear equation (depicted in chart bottom right). Note that the coefficient in the equation is significantly different than the averaged percentages (and even the individual percentages), due to a noticeable non-zero y-intercept.

Remember, IV&V requirements vary by program. The additional cost of this activity provides assurance beyond just in-house testing.

SW Devel	IV&V	IV&V %
$ 10.0	$ 1.0	10.0%
$ 50.0	$ 2.5	5.0%
$ 30.0	$ 2.0	6.7%
$ 25.0	$ 1.8	7.2%
$ 40.0	$ 2.8	7.0%
$ 31.0	**$ 2.0**	**7.2%**

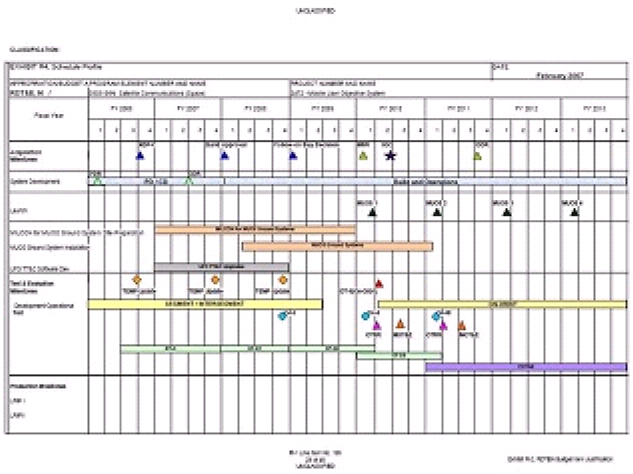


Schedule Estimating Relationships (SERs)

For acquisition programs in general and for software development in particular, it is important to consider schedule in conjunction with cost.

Using estimates of cost and effort, a tentative, projected schedule is developed. While detailed network schedules for a project are usually the purview of the implementers of the earned value management system (EVMS), schedule estimating, especially using parametric techniques to predict and assess top-level schedules, often falls to the cost analyst.

The cost analyst is intimately familiar with these techniques and has often collected the cost and schedule data needed to drive them.



Questions Managers Should Ask

Popup Text

Questions Managers Should Ask

Have steps been taken to ensure the integrity of the estimating process?

- Are the cost and schedule estimated consistently?

Schedule Estimating Relationships (SERs), Cont.

A schedule estimating relationship (SER) is used to predict schedule (duration) for a project in the same way that a CER is used to predict its cost (effort).

Data from several comparable historical programs are used to derive an equation (preferably using regression analysis) that relates the dependent variable of Schedule to one or more independent variables.

As is the case with the COCOMO II, the primary driver for a software development SER is often total effort. In this case, the SER operates very much like the Cost-on-Cost CERs just discussed: its input parameter is actually the output from a related estimate.

Whereas effort is measured in units (such as person-months or labor hours) that account for multiple developers working concurrently, schedule is measured in units of literal time: days, months, or years.

To emphasize this distinction, one might refer to schedule units as calendar months or calendar years (instead of person-months or staff-years).

Popup Text

Schedule Estimating Relationship (SER)

A parametric relationship that estimates the total schedule (duration) of a program, project, or task based on historical data from several comparable efforts, often driven by the corresponding total labor hours or cost.

Schedule Estimating Relationships - COCOMO SER

The graphic illustrates a slightly simplified version of the primary COCOMO II SER, a function of effort in person-months (PM) as estimated by the COCOMO II CER.

Its scale factors have also been set to reflect modest diseconomies of scale, as captured by the 1.1 exponent in the CER.

The exponent of the SER is higher as a result of those diseconomies of scale, though it is still significantly less than 1, indicating that schedule (in calendar months) grows, but at a decreasing rate, as effort (in person-month) grows.

$$TDEV = 3.67 \cdot PM^{0.32}$$

Schedule Estimating Relationships - Schedule Compression

It is not unusual for there to be a specific schedule imposed on a software development effort. The deadline may be driven by an external need such as Y2K, or it may simply be that the contractor proposed a certain date, and it will cost money if it is missed.

No program is immune to schedule pressures, but software projects are arguably more susceptible, because the product (and hence progress thereon) is more intangible, and there may be a greater temptation to "declare victory" and deliver as is.

Compressing a schedule to less than a normal length of time can require more programmers on the effort leading to more opportunity for failed communication and coordination.

It may also cause developers to cut corners – to fail to spend the necessary time in understanding the requirements and laying out the design, which can lead to problems discovered during coding and unit testing.

Schedule Estimating Relationships - Schedule Compression, Cont.

Fixes made during code and test normally take much longer and require more effort than problems that are found during the design phase.

Programmers may not document or test the code as well as they would if they were not under schedule constraints which can lead to problems in maintenance and ease of reuse.

Though not evident in the simplified COCOMO II CER, a compressed schedule is expected to result in greater total effort (PM), not just the increased staffing levels that are an obvious consequence of dividing the same effort over a shorter duration.

There is a schedule-related Effort Multiplier (EM) that captures this effect. In COCOMO II, there is no penalty for lengthening a project beyond its nominal schedule, though other sources show this should also increase total effort but not as acutely.

Time-Phasing

In conjunction with estimating the effort (total labor hours or person-months) and the schedule (calendar days or months), it is generally good practice to spread or "time-phase" the effort across the schedule.

Initially, a broad-brush, top-level approach will suffice, in support of annual program budgets. For contract execution, a more detailed, bottom-up approach is desired to produce a month-by-month allocation.

Two key cross-checks for time-phasing are average staffing and peak staffing, both usually expressed in full-time equivalents (FTE).

If effort and schedule estimates are already in person-months (calibrated to the appropriate standard hours per month) and calendar months, respectively, then the quotient of the two should be the average staffing in FTE. The peak staffing is dependent upon the time-phasing and is often significantly higher than the average.

Can they hire that many qualified developers (peak)? Can they maintain that level of staffing (average)? If the answer to either of these is no, consider extending the schedule or adjusting the time-phasing.

NOTE

Popup Text

Note

DAU offers a continuous learning module, CLB031 Time Phasing, with much more detail and applications beyond just software estimating.

Time-Phasing - Time-Phasing Techniques

As with effort and schedule, time-phasing may apply Analogy and Parametric methodologies. A Resource-Loaded Schedule approach may also be used. Select each tab below to read more.

| Parametric | Resource-Loaded Schedule |

Apply (empirical) time-phasing profiles from actual data on a similar project.

Often these profiles are in percentages so that they can be applied across varying durations.

This can be done at a total project level, or different profiles can be applied to different activities (requirements, development, testing, etc.).

As historical data are being gathered, "anecdotal actuals" in the form of rough percentages from a SME may do as a first cut.

Popup Text

Analogy

Apply (empirical) time-phasing profiles from actual data on a similar project.

Often these profiles are in percentages so that they can be applied across varying durations.

This can be done at a total project level, or different profiles can be applied to different activities (requirements, development, testing, etc.).

As historical data are being gathered, “anecdotal actuals” in the form of rough percentages from a SME may do as a first cut.

Parametric

Similar to the analogy approach, but the historical data are abstracted to a best-fit probability distribution based on one or more actual projects.

The cumulative distribution function (CDF) or S-curve shows the cumulative percent complete from 0% to 100% as a function of elapsed time, which can be scaled to any estimated duration.

Common distributions used include Rayleigh, Weilbull, and Beta.

Popup Text

Resource-Loaded Schedule

The performance measurement baseline, or PMB, used to implement earned value management (EVM) on a project is essentially a resource-loaded schedule, or a time-phased budget.

The nuanced difference between the two is that the former generally entails that specific resources, i.e., developers, not just labor categories, have been assigned to tasks.

Picture a Gantt chart depicting a series of interrelated tasks and milestones, with the best estimates of when each task will commence, the associated effort and duration, and any predecessor/successor dependencies.

The associated hours may be time-phased within each task, using the Analogy or Parametric approaches just discussed. If the task is relatively short or low level of effort, a Uniform distribution (even spread or “level loading”) may suffice.

Time-Phasing - Schedule Building and QA

Resource-Loading may also be used to determine the schedule (duration) itself, either as a cross-check of a parametric SER (early on) or as the primary methodology (later, during execution).

One or more software engineers with experience in the specific application under development should develop a schedule estimate as follows:

- Expand the WBS to delineate the order in which functional elements will be developed. The order of development will define which functions can be developed in parallel as well as dependencies that drive the schedule.
- A development schedule should be derived for each set of functions that can be developed independently, for example, a schedule for each build of an incremental development.
- The schedule for each set of independent functions should be derived as the aggregate of the estimated time required for each major phase of the development: requirements analysis; design, code, and unit test; and integration and test.
- The total project schedule should reflect the aggregate of the product development, including documentation and formal review requirements.

Program schedules are often fraught with errors, inconsistencies, and insufficient estimates. It is generally wise to apply independent criteria to assess the adequacy of a schedule. This can be done with a combination of automated models and manual review by software and schedule experts. The GAO Schedule Assessment Guide may prove helpful.

Off-The-Shelf (OTS) Cost Models

Several ready-made cost estimating models, referred to as off-the-shelf, or OTS, models, can assist in developing software estimates, offering a graphical user interface (GUI) and other conveniences.

These models have been developed over the years with industry data from hundreds, even thousands, of projects, and they can be tailored and calibrated to specific program design requirements or program actuals.

Some of the most commonly used models in the industry are discussed in the lesson. More information is available on the vendors' websites.

COCOMO II is used as an example throughout this module because of its status as a widely-used, widely-scrutinized academic model, with some insight into how its estimating relationships are derived from actual data.

QSM®

AIR FORCE COST ANALYSIS AGENCY

PRICE
True Program Success™

SEER® by GALORATH

Questions Managers Should Ask

Popup Text

Questions Managers Should Ask

Are the estimated costs and schedule consistent with demonstrated accomplishments on other projects?

- Were any cost and schedule models used to develop the cost estimate calibrated to relevant historical data?
- Was the calibration of any cost and schedule models done with the same versions of the models that were used to prepare the estimate?

Have steps been taken to ensure the integrity of the estimating process?

- Is at least one member of the estimating team an experienced estimator and trained in any cost models that were used?

Off-The-Shelf (OTS) Cost Models, Cont.

OTS cost models generally require little or no data, so they are useful when no historical data are available.

However, if you use one of these models with no historic data specific to your type of program or environment, you're implicitly accepting the "generic" estimate produced by the model based on industry-average data. It is generally preferable to calibrate OTS models.

The primary disadvantage of using these models is the so-called "black-box" syndrome, the limited insight into the processes applied to derive the estimate, such as underlying data sets, component CERs, statistical significance, and the like.

Off-The-Shelf (OTS) Cost Models - Functionality

The OTS models generally have a number of inputs that attempt to characterize the software project and its relative scope and difficulty, including:

- Sizing and reuse information
- Developer tools and experience
- Software application and quality requirements

Examples of additional environmental parameters that must be determined and entered into the model include, but are not limited to, program complexity, programming language, requirements volatility, analyst capability, and execution time constraint. Many of these are similar to the COCOMO effort multipliers (EMs) seen earlier.

Based on these inputs, the models will determine total effort, schedule, and time-phasing. As noted, this is a "black box," with the algorithms used to do so not completely documented. To some extent, you can "reverse-engineer" a model by varying input parameters and seeing how the corresponding outputs change.

OTS models may provide a variety of outputs, including estimate by WBS and various summary graphs. The risk and uncertainty capabilities of most OTS models is limited, and often risk analysis needs to be conducted in a separate model.

Off-The-Shelf (OTS) Cost Models - Calibration and Cross-Checks

When using an OTS model, attempt to calibrate it using actual cost, technical, and programmatic data from at least one comparable historical program.

The model is adjusted to more closely reproduce the historical costs when given the corresponding parameter values as inputs.

Beware that the model calibration process is often "fuzzy" and must be done with care and well documented so as to be defensible.

OTS software models, given equivalent inputs, would be expected to produce results of at least the same order of magnitude, but this is not always the case.

As with any cross-check, if the results of two different methods are comparable, confidence in the estimate increases.

Both cost (effort) and schedule (duration) estimates should be compared.

D

Long Description

Semispherical gauge illustration named Legacy Program with Cost, Technical and Data located at different points of the gauge face.

Off-The-Shelf (OTS) Cost Models - Calibration and Cross-Checks, Cont.

If two models produce significantly different estimates, it calls into question whether both have been thoroughly calibrated and tested.

If this happens, confidence in the estimate decreases pending further investigation.

Cross-checks focus attention on the content of the estimate and the techniques used to derive it.

Due to the inherent uncertainty of all estimating methods, never expect two methods to produce precisely the same result but rather to be "in the same ballpark."

Off-The-Shelf (OTS) Cost Models - Market Survey

Each of the following OTS Cost Models can be used for estimation purposes. Select each tab to learn more.

	True S®	SEER-SEM®	SLIM-Estimate®

The Revised Enhanced Version of Intermediate COCOMO, or REVIC, was developed by Mr. Ray Kile, an estimator at the Air Force Cost Analysis Agency (AFCAA) in the 1980s.

He felt that COCOMO was not specific enough for DoD use, so he revised the COCOMO database.

The main difference between REVIC 94 and COCOMO II are the coefficients used in the effort equation, which in REVIC are based on data from only DoD projects.

Select the AFCAA seal for more information.

Popup Text

REVIC 94

The Revised Enhanced Version of Intermediate COCOMO, or REVIC, was developed by Mr. Ray Kile, an estimator at the Air Force Cost Analysis Agency (AFCAA) in the 1980s.

He felt that COCOMO was not specific enough for DoD use, so he revised the COCOMO database.

The main difference between REVIC 94 and COCOMO II are the coefficients used in the effort equation, which in REVIC are based on data from only DoD projects.

Select the AFCAA seal for more information.

True S®

Formerly known as PRICE S®, the PRICE Systems software model is part of the True Planning suite of estimating tools.

It is used for estimating the costs and schedules of software development projects.

Output includes effort in person-months or dollars and schedule in a report format that is highly tailorable.

Select the logo to learn more about the PRICE Systems.

Popup Text

SEER-SEM®

Software Evaluation and Estimation of Resources Software Estimating Model (SEER-SEM) predicts, measures, and analyzes resources, staffing, schedules, and costs for software projects. Outputs include effort in person-months or dollars and schedule in a variety or report formats.

SEER-SEM can be calibrated by computing an effective technology rating (ETR) from past programs. The ETR is one of the factors used by SEER-SEM in processing. The model is also tailorable for different labor rates, phases, etc. Select the logo below to view the Galorath website for more information.

SLIM-Estimate®

Part of the Software Lifecycle Management (SLIM suite), SLIM-Estimate helps you estimate the time, effort, and cost required satisfying a given set of requirements and determining the best strategy for designing and implementing your software or systems project.

Select the logo to view more information on the QSM website.

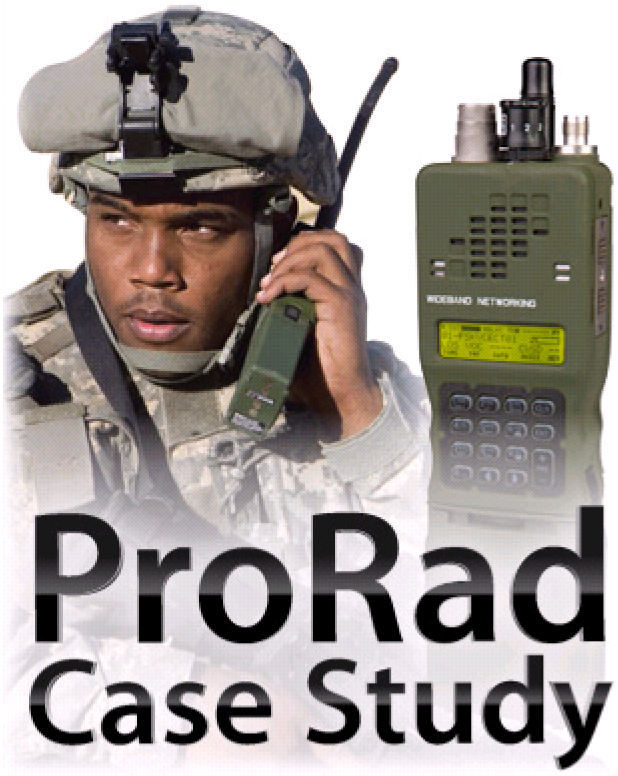
ProRad Estimates

The Joint ProRad Program Office used an OTS software model to estimate the software development effort for the 31 waveforms. Inputs included:

- User requirements
- Lines of code to develop, verify, and test
- Character of the code: reusable/new; designed for reuse; modified or new object design and develop
- Character of the development team: experience; skill
- Minimum time constraints
- Development and target host environments
- Development and integration risk

The cost for each waveform was calculated as: Effort (in Staff Months) x Average Staff Month Rate (Loaded).

The resulting costs were distributed over the development schedule and escalated to then-year dollars as shown in this table.

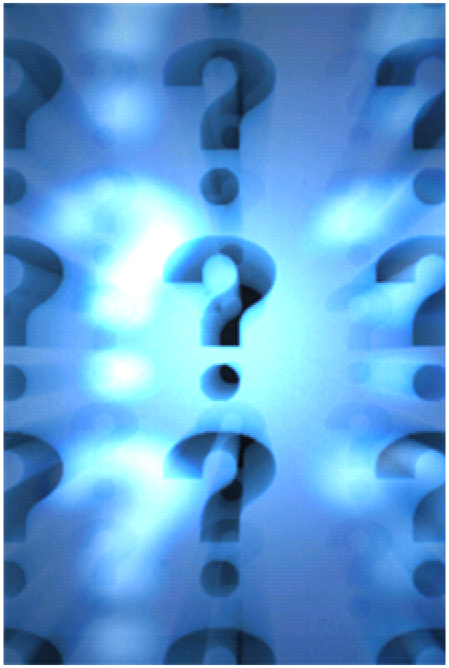


Knowledge Review

FILL IN THE BLANKS: The simplified COCOMO II CER applied to new code development gives effort in ____ as a function of size in ____.

- ☐ Labor hours; SLOC
- ☐ Labor hours; KSLOC
- ☐ Person-months; SLOC
- ✓ Person-months; KSLOC

Check Answer



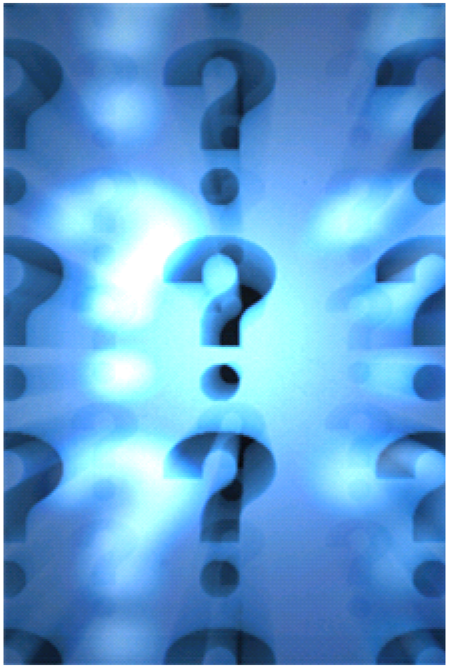
The simplified COCOMO II CER applied to new code development gives effort in **person-months** as a function of size in **KSLOC**.

Knowledge Review

FILL IN THE BLANKS: The simplified COCOMO II SER gives schedule in _____ as a function of effort in _____.

- ☐ Calendar days; labor hours
- ☐ Calendar days; person-months
- ☐ Calendar months; labor hours
- ✓ Calendar months; person-months

Check Answer



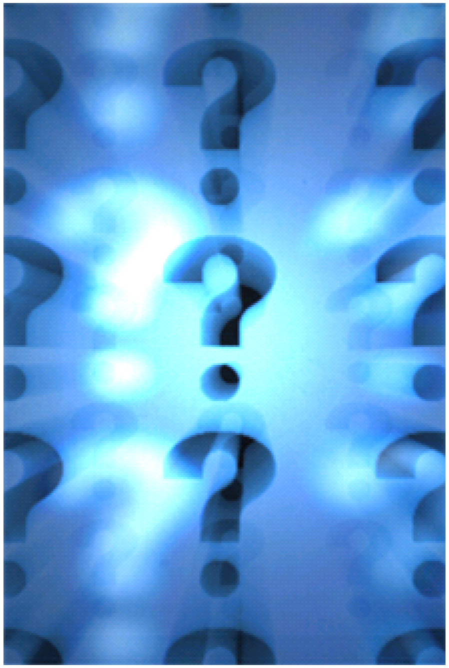
The simplified COCOMO II SER gives schedule in **calendar-months** as a function of effort in **person-months**.

Knowledge Review

Which of the following is **not** an advantage of Parametric over Analogy as a technique for software estimates.

- ☐ Has an objective measure of validity (statistical significance)
- ✓ Requires less data (one historical program vs. many)
- ☐ Has a statistical measure of uncertainty (standard error)
- ☐ Can account for economies or diseconomies of scale (non-linear)

Check Answer



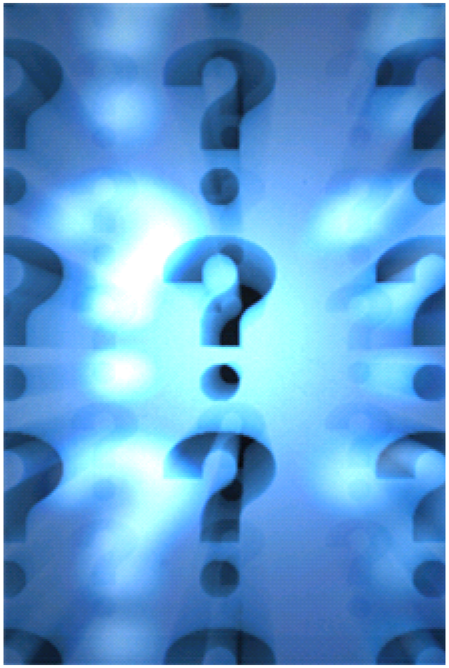
Requiring less data is not an advantage of Parametric over Analogy as a technique for software estimates. On the contrary, Parametric requires several comparable historical data points, whereas a classic Analogy can be achieved with only one.

Knowledge Review

Which of the following is **not** typically true of OTS software estimating models?

- ☐ Evolved over many years
- ☐ Based on industry data
- ☑ Lack a graphical user interface (GUI)
- ☐ Estimate both cost (effort) and schedule (duration)
- ☐ Have multiple input parameters

Check Answer

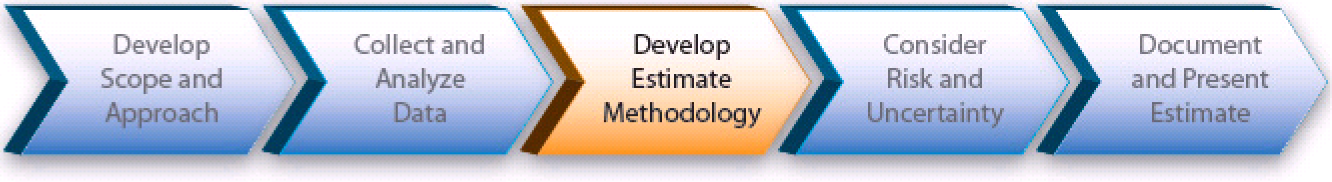


Lack a graphical user interface (GUI) is not typically true of OTS software estimating models.

Summary

This completes the Develop Estimate Methodology lesson. In this lesson, you learned:

- Parametric is the preferred technique for software estimate, while Analogy is useful for sizing and when comparable historical data are limited.
- Cost estimating relationships (CERs) and schedule estimating relationships (SERs) are equations for effort and duration, respectively, with inputs corresponding to the drivers.
- Software estimates may need to be time-phased to support budgeting (usually annual) and planning (usually monthly).
- Many off-the-shelf (OTS) models for software estimating are available, but they should be calibrated to your data and used with caution.


Develop Scope and Approach
Collect and Analyze Data
Develop Estimate Methodology
Consider Risk and Uncertainty
Document and Present Estimate

D

Long Description

Graphic illustrates the steps of the Cost Estimating process. The steps from left to right are:
Develop Scope and Approach, Collect and Analyze Data, Develop Estimate Methodology (highlighted), Consider Risk and Uncertainty, and Document and Present Estimate.

Lesson Completion

You have completed the content for this lesson.

To continue, select another lesson from the Table of Contents on the left.

If you have closed or hidden the Table of Contents, click the Show TOC button at the top in the Atlas navigation bar.