Overview of COCOMO

The COCOMO cost estimation model is used by thousands of software project managers, and is based on a study of hundreds of software projects. Unlike other cost estimation models, COCOMO is an open model, so all of the details are published, including:

- The underlying cost estimation equations
- Every assumption made in the model (e.g. "the project will enjoy good management")
- Every definition (e.g. the precise definition of the Product Design phase of a project)
- The costs included in an estimate are explicitly stated (e.g. project managers are included, secretaries aren't)

Because COCOMO is well defined, and because it doesn't rely upon proprietary estimation algorithms, Costar offers these advantages to its users:

- COCOMO estimates are more objective and repeatable than estimates made by methods relying on proprietary models
- COCOMO can be calibrated to reflect your software development environment, and to produce more accurate estimates

Costar is a faithful implementation of the COCOMO model that is easy to use on small projects, and yet powerful enough to plan and control large projects.

Typically, you'll start with only a rough description of the software system that you'll be developing, and you'll use Costar to give you early estimates about the proper schedule and staffing levels. As you refine your knowledge of the problem, and as you design more of the system, you can use Costar to produce more and more refined estimates.

Costar allows you to define a software structure to meet your needs. Your initial estimate might be made on the basis of a system containing 3,000 lines of code. Your second estimate might be more refined so that you now understand that your system will consist of two subsystems (and you'll have a more accurate idea about how many lines of code will be in each of the subsystems). Your next estimate will continue the process -- you can use Costar to define the components of each subsystem. Costar permits you to continue this process until you arrive at the level of detail that suits your needs.

One word of warning: It is so easy to use Costar to make software cost estimates, that it's possible to misuse it -- every Costar user should spend the time to learn the underlying COCOMO assumptions and definitions from ***Software Engineering Economics*** and ***Software Cost Estimation with COCOMO II***.

Introduction to the COCOMO Model

The most fundamental calculation in the COCOMO model is the use of the Effort Equation to estimate the number of Person-Months required to develop a project. Most of the other COCOMO results, including the estimates for Requirements and Maintenance, are derived from this quantity.

Source Lines of Code

The COCOMO calculations are based on your estimates of a project's size in Source Lines of Code (SLOC). SLOC is defined such that:

- Only Source lines that are DELIVERED as part of the product are included -- test drivers and other support software is excluded
- SOURCE lines are created by the project staff -- code created by applications generators is excluded
- One SLOC is one logical line of code
- Declarations are counted as SLOC
- Comments are not counted as SLOC

The original COCOMO 81 model was defined in terms of Delivered Source Instructions, which are very similar to SLOC. The major difference between DSI and SLOC is that a single Source Line of Code may be several physical lines. For example, an "if-then-else" statement would be counted as one SLOC, but might be counted as several DSI.

The Scale Drivers

In the COCOMO II model, some of the most important factors contributing to a project's duration and cost are the Scale Drivers. You set each Scale Driver to describe your project; these Scale Drivers determine the exponent used in the Effort Equation.

The 5 Scale Drivers are:

- Precedentedness
- Development Flexibility
- Architecture / Risk Resolution
- Team Cohesion
- Process Maturity

Note that the Scale Drivers have replaced the Development Mode of COCOMO 81. The first two Scale Drivers, Precedentedness and Development Flexibility actually describe much the same influences that the original Development Mode did.

Cost Drivers

COCOMO II has 17 cost drivers – you assess your project, development environment, and team to set each cost driver. The cost drivers are multiplicative factors that determine the effort required to complete your software project. For example, if your project will develop software that controls an airplane's flight, you would set the Required Software Reliability (RELY) cost driver to Very High. That rating corresponds to an effort multiplier of 1.26, meaning that your project will require 26% more effort than a typical software project.

Click here to see which Cost Drivers are in which Costar models.

COCOMO II defines each of the cost drivers, and the Effort Multiplier associated with each rating. Check the Costar help for details about the definitions and how to set the cost drivers.

COCOMO II Effort Equation

The COCOMO II model makes its estimates of required effort (measured in Person-Months – PM) based primarily on your estimate of the software project's size (as measured in thousands of SLOC, KSLOC)):

$$\text{Effort} = 2.94 * \text{EAF} * (\text{KSLOC})^{E}$$

Where

EAF Is the Effort Adjustment Factor derived from the Cost Drivers

E Is an exponent derived from the five Scale Drivers

As an example, a project with all Nominal Cost Drivers and Scale Drivers would have an EAF of 1.00 and exponent, E, of 1.0997. Assuming that the project is projected to consist of 8,000 source lines of code, COCOMO II estimates that 28.9 Person-Months of effort is required to complete it:

$$\text{Effort} = 2.94 * (1.0) * (8)^{1.0997} = 28.9 \text{ Person-Months}$$

Effort Adjustment Factor

The Effort Adjustment Factor in the effort equation is simply the product of the effort multipliers corresponding to each of the cost drivers for your project.

For example, if your project is rated Very High for Complexity (effort multiplier of 1.34), and Low for Language & Tools Experience (effort multiplier of 1.09), and all of the other cost drivers are rated to be Nominal (effort multiplier of 1.00), the EAF is the product of 1.34 and 1.09.

Effort Adjustment Factor = EAF = 1.34 * 1.09 = 1.46

Effort = 2.94 * (1.46) * $(8)^{1.0997}$ = 42.3 Person-Months

COCOMO II Schedule Equation

The COCOMO II schedule equation predicts the number of months required to complete your software project. The duration of a project is based on the effort predicted by the effort equation:

Duration = 3.67 * $(\text{Effort})^{SE}$

Where

Effort Is the effort from the COCOMO II effort equation
SE Is the schedule equation exponent derived from the five Scale Drivers

Continuing the example, and substituting the exponent of 0.3179 that is calculated from the scale drivers, yields an estimate of just over a year, and an average staffing of between 3 and 4 people:

Duration = 3.67 * $(42.3)^{0.3179}$ = 12.1 months

Average staffing = (42.3 Person-Months) / (12.1 Months) = 3.5 people

The SCED Cost Driver

The COCOMO cost driver for Required Development Schedule (SCED) is unique, and requires a special explanation.

The SCED cost driver is used to account for the observation that a project developed on an accelerated schedule will require more effort than a project developed on its optimum schedule. A SCED rating of Very Low corresponds to an Effort Multiplier of 1.43 (in the COCOMO II.2000 model) and means that you intend to finish your project in 75% of the optimum schedule (as determined by a previous COCOMO estimate). Continuing the example used earlier, but assuming that SCED has a rating of Very Low, COCOMO produces these estimates:

Duration = 75% * 12.1 Months = 9.1 Months

Effort Adjustment Factor = EAF = 1.34 * 1.09 * 1.43 = 2.09

Effort = 2.94 * (2.09) * $(8)^{1.0997}$ = 60.4 Person-Months

Average staffing = (60.4 Person-Months) / (9.1 Months) = 6.7 people

Notice that the calculation of duration isn't based directly on the effort (number of Person-Months) – instead it's based on the schedule that would have been required for the project assuming it had been developed on the nominal schedule. Remember that the SCED cost driver means "accelerated from the nominal schedule".

The Costar command ***Constraints | Constrain Project*** displays a dialog box that lets you trade off duration vs. effort (SCED is set for you automatically). You can use the dialog box to constrain your project to have a fixed duration, or a fixed cost.

A Brief History of COCOMO

1981

The original COCOMO is introduced in Dr. Barry Boehm's textbook ***Software Engineering Economics***. This model is now generally called "COCOMO 81".

1986

Costar 1.0 is released.

1987

Ada COCOMO and Incremental COCOMO are introduced (proceedings, Third COCOMO Users Group Meeting, Software Engineering Institute).

1988, 1989

Refinements are made to Ada COCOMO.

Costar 2.0 is released.

1990

Costar 3.0 is released.

1993

Costar 4.0 is released.

1995, 1996

Early papers describing COCOMO 2 published.

1997

The first calibration of COCOMO II is released by Dr. Boehm, and named "COCOMO II.1997".

Costar 5.0 is released, supporting COCOMO II.

1998

The second calibration of COCOMO II is released. It's named "COCOMO II.1998".

1999

COCOMO II.1998 is renamed to COCOMO II.1999 and then to COCOMO II.2000 (all three models are identical).

2000

The book ***Software Cost Estimation with COCOMO II*** (Dr. Barry Boehm, et al) is published to document how to apply the latest estimation model. Most of the original ***Software Engineering Economics*** is still applicable to modern software projects.

Costar 6.0 is released, and supports COCOMO II.2000, MBASE, REVIC, and schedule constraints.

2003

Costar 7.0 is released, and supports drag & drop, wizards, XP, import from USC tool, toolbars, and filters.

Costar is the only estimation tool that supports **all** of the COCOMO estimating models!

Ada COCOMO

COCOMO has continued to evolve and improve since its introduction. Costar supports the traditional COCOMO model, and the most recent models.

The document TRW IOC Ada COCOMO: Definition and Refinements (Barry Boehm & Walker Royce, 1987, 1988) defines the "Ada COCOMO" model.

Each model consists of cost drivers, equations, phase distribution tables, etc., and is stored in a file as a Costar model (with extension "mdl"). You can modify any of the Costar models to match your development environment. Costar 7.0 has 13 built-in models.

The two cost drivers, SECU (Classified Security Application) and RUSE (Required Reusability) have been added to the Ada COCOMO model.

The model named "ADA_87" assumes that the Ada programming language is being used. Use of Ada makes it easier to develop highly reliable systems and to manage complex applications, so the cost drivers RELY and CPLX have new definitions.

The APM_88 model is based upon a different "process model" called the Ada Process Model. Among the assumptions are:

- **You use Ada to produce compiler checked package specifications by the Product Design Review (PDR)**
- **Small design teams are used**
- **More effort is spent in requirements analysis and design**
- **Less effort is spent in coding, integration, and testing**

The APM_88 model incorporates changes to the equations, several cost drivers, and a variety of other tables.

This table summarized the different COCOMO models you can use in Costar. Remember that you can use our tools to create your own versions of the COCOMO models and tables, calibrated to your development environment.

Model	Ada Language?	Equations	Description
COCOMO_85	No	Traditional	*Software Engineering Economics* + TURN VL, TOOL XH
COCOMO_87	No	Traditional	New VMVH, VMVT, SECU, RUSE, SCED
ADA_87	Yes	Traditional	New RELY, CPLX, LEXP
APM_88	Yes	Ada Process	New MODP, ACAP, PCAP, sigma, tables...

Incremental COCOMO

Incremental Development COCOMO was defined at the same time as Ada COCOMO. Incremental COCOMO is a modern alternative to the traditional Waterfall model of the software development process.

Incremental Development COCOMO lets you model a variety of development processes. Instead of modeling your software development as if it were a single effort devoted to inventing a single product, Incremental Development COCOMO lets you model development as a series of concurrent software projects, each yielding an intermediate product.

This strategy reduces your risk, and permits you to deliver an initial product to your customer earlier.

We've extended and generalized the definition so that Costar performs the calculations for increments with multiple components.

You can use Costar's worksheets to define which development phases are included in each increment, and how each increment is synchronized with the other increments. You can add delays between phases or between increments to match your schedule.

You can assign any component to any increment.

Incremental Development can be used with any COCOMO model, whether it's Ada COCOMO, or standard COCOMO. It can be used with Organic, Semidetached, or Embedded projects.

Function Points

The Function Point methodology was developed by Allan Albrecht at IBM.

The Function Point methodology is based on the premise that the size of a software project can be estimated early, during the requirements analysis, based upon the inputs and outputs of the system. Five classes of items are counted:

1. **External Inputs**
2. **External Outputs**
3. **Logical Internal Files**
4. **External Interface Files**
5. **External Inquiries**

Based upon counts for each of these items, and the weighting factors and adjustment factors that Albrecht proposes, you can calculate a Total Function Point count.

Costar converts the Function Point count into an equivalent number of SLOC, and uses that in the COCOMO equations to make its estimates.

Using Costar

Most of your interactions with Costar will involve creating and modifying components. You'll define subcomponents, assign cost driver values, estimate the size of each component, etc.

One component is always distinguished as "the current component". The name of the current component, the cost drivers you've assigned to it, and other data describing the current component are shown in the main Costar window.

A given component may be made up of any number of subcomponents. When you create a new subcomponent, it becomes a subcomponent of the current component. It inherits values for each of its attributes from its parent component. The following attributes are inherited:

- **Settings for the Cost Drivers**
- **Cost per Person-Month settings**
- **Maintenance, Adaptation, and Conversion settings**

Costar makes it easy to perform "what-if" analyses, and to compare different project plans. You may develop a new estimate based upon an older one, and then use Costar to compare the two.

A Costar estimate consists of:

- **A name**
- **An ID**
- **A one line comment**
- **5 Scale Drivers (COCOMO II models) or a Development Mode (COCOMO 81 models)**
- **An associated database (e.g. COCOMO II.2000)**
- **APM settings**
- **Names & costs of labor classes**
- **One or more components**

One estimate is always distinguished as "the current estimate". All of the Costar commands operate on the current estimate.

When there is only a single component in an estimate, the SLOC value for the component is identical to the total SLOC value for the estimate. But when a component has subcomponents, the SLOC value is derived from the SLOC values of its subcomponents.

Costar lets you specify SLOC values in three different ways:

1. **You can explicitly enter a SLOC value such as 3,000.**
2. **You can use the Reuse Tab to calculate the SLOC.**
3. **You can use the Function Point Tab to calculate the SLOC.**

==

Table of Contents

---

1. What's the definition of a Person-Month?

A Person-Month is one month of effort by one person. In the olden days, a Person-Month would have been called a Man-Month or a Staff-Month.

In standard COCOMO, there are exactly 152 hours per Person-Month. In your organization, you might have a definition that differs from the standard by 10% to 20%.

Luckily, the Calico program that is included with Costar lets you redefine the number of Hours per Person-Month to match your organization's definition, so this is taken care of automatically by Costar.

If you're doing the COCOMO calculations by hand, or using one of the free versions of COCOMO, you must make this adjustment manually, or your estimates will be wrong.

Back to Top

---

2. Should I estimate 2 related projects together?

It depends.

Consider the form of the estimating equations (as explained in the COCOMO Overview). The Effort Equation has an exponent greater than 1, indicating that software projects experience a diseconomy of scale. That reflects the common experience that a 10,000 line project will take more effort than 10 projects of 1,000 lines each. On a larger project, more effort is spent on communication and organization, rather than on technical work.

So, if your projects will proceed without much interaction between them, you should estimate them as 2 separate projects. The diseconomy of scale won't apply.

But, if the 2 projects will be coordinated with each other, requiring a lot of contact and interaction with each other, you should consider them as pieces of a single large project, and make a single estimate that includes both.

Note that Costar lets you decompose an estimate into any number of components and subcomponents, so you can model your estimate as having 2 major components if you like. Each of those components might have dozens (or hundreds) of subcomponents.

Back to Top

---

3. How should I handle a Year 2000 estimate?

Costar supports the COCOMO Reuse & Adaptation calculations. After reviewing a sample of your code, you should be able to supply the following figures for your system:

1. How large is the system?

   Let's work an example assuming the system is 1 million Source Lines of Code (SLOC).

2. How much of the design will you need to modify?

   Let's assume that you'll change just 2% of the design.

3. How much of the code will you need to modify?

Let's assume that you'll change 5% of the code.

4. How much Integration & Testing will be performed on the system?

   We'll assume that the testing will be just as thorough as it was during the original development.

Using the Costar Reuse Tab, you'll arrive at an Equivalent SLOC (323,000 in this example). That's the figure that will be pushed through the standard COCOMO estimating equations to make estimates for your project.

When you try this, you may be shocked at the size of the effort predicted. There are a couple of factors working in your favor, though. You can probably set the Precedentedness scale driver to "Thoroughly Familiar" and Architecture/Risk Resolution to 100% because is a well understood problem being undertaken in a familiar environment. You can probably assume that the Complexity cost driver is very low, and that the Applications Experience cost driver is very high -- setting these 2 cost drivers will trim the estimated effort by over 40%.

Back to Top

---

4. How can I estimate the value of my software company?

There are many ways to estimate the value of a software company, but none of them are very good. Costar and COCOMO can help you estimate the value of the software itself (which might be *much* less than the value of the company). The simplest approach is to estimate the "replacement cost" -- what would it cost someone else to duplicate your product from scratch?

The folks at The Corum Group are experts at valuing software companies.

Back to Top

---

5. How do I calibrate Costar to my environment?

Out-of-the-box, Costar estimates pretty well for most organizations, but in an ideal world, you'd tune it to match your organization's history.

Here's the hard part: collect data describing completed projects (size, effort, duration, cost drivers). You'll need at least half a dozen data points to get reasonable results.

Here's the easy part: type your data into Calico, select a style of calibration, and press Enter. Calico is our COCOMO calibration tool -- it's included with every

purchase of Costar. Calico takes a description of your completed projects as inputs, and calibrates the COCOMO equations to match your environment. You can arrange for Costar to automatically use your new equations instead of the standard estimating equations.

No, Calico doesn't use magic -- it turns out that all you need is a linear regression.

Back to Top

---

6. How many different inputs are there?

Somewhere between 1 and 1,000,000 different inputs.

At a minimum, you need to describe your project with a size in Source Lines of Code (SLOC) or Function Points, so you need at least one input (!).

But, for each component, there are 50 to 100 different items you can set (costs, cost drivers, maintenance cost drivers, etc.), and in Costar 7.0 you can create thousands of components, so in theory, there are hundreds of thousands of possible inputs.

In practice, you'll probably set only a handful of parameters for each component.

Subcomponents in Costar can inherit attributes from their parent components, so if you set the Programmer Capability cost driver to Very High for the top level component, that value will be inherited by all subcomponents (unless overridden by a different value in a particular subcomponent). That simplifies life.

Back to Top

---

7. When should I use the Incremental Development model?

They key word here is "model". You use Costar and COCOMO to create a model of a software project, then you press a button and the program does the calculations.

If you plan to develop your software in 2 or more increments, use the Incremental Development model to do the estimates. Using Costar, it's simple to assign a component to any one of your increments -- and if you want to switch back to a single increment, just assign all of your components to increment 1.

Back to Top

---

8. How should I model a Rapid Prototyping project?

You could model that as a series of small increments, where each increment adds more functionality to the product.

We've generalized the usual Incremental COCOMO a little bit to handle this sort of situation. In Costar, you can use the Phasing Worksheet to describe how each of your increments overlaps with the other increments (you may have several going on at once).

In the traditional "waterfall" model of a software project, we pretended that all of the requirements and design was done first, before we started any coding and testing.

In a project with a lot of prototyping and concurrent development, you won't be doing all of the Requirements and Product Design at the beginning of the project -- you'll be doing some of each throughout the project as you learn more about what it is you're developing. In Costar, you can use the Phasing Worksheet to model that approach -- you can specify that you'll be doing some Requirements and Design work at the beginning of every increment -- not just before the first one.

You can also use Costar's Increment Breakage worksheet to describe the amount of code you'll be throwing away in each increment (the "breakage"). There will be some breakage because you'll probably have to abandon some ideas and features as you home in on what it is you're trying to build. If the breakage is a large number (say, 50%), you've slipped over the edge from Rapid Prototyping to Rabid Prototyping.

Back to Top

---

9. Does COCOMO II replace traditional COCOMO?

You should use COCOMO II for most of your new projects.

COCOMO 81 is still the best approach for some software projects. If you're using a fairly traditional approach, and using a 3GL (third generation language), such as C, Fortran, or Cobol, the original COCOMO will give you good results. If your development tools and processes haven't changed much in recent years, COCOMO 81 might be the right model for you.

If you're using more modern tools, a new development process model, a 4GL language, or tools like Visual Basic, Delphi, or Power Builder, you might find that COCOMO II makes more sense.

When in doubt, try to model your project using both the traditional COCOMO and COCOMO II.

Back to Top

---

10. What version of Function Points do you support?

Do you remember the episode of M*A*S*H in which Col. Potter refers to the Great War, and says "...that was before we knew enough to start numbering them..."?

Well, we use the original Albrecht definitions, before any names or version numbers were attached to Function Points.

But, just about everything in Costar can be customized, one way or another. Using the Calico tool included with Costar, you can change all of the FP weights, and invent your own version of function points. Just don't tell the people at IFPUG about it.

Back to Top

---

11. When is COCOMO II.2000 being released?

The Post-Architecture Model and the Early Design Model have been released. The Applications Composition Model is still under development at USC.

Our Costar 7.0 for Windows supports the most recent changes to COCOMO II.

Back to Top

---

12. What do I use Calico for?

We like to think of Costar and Calico as a suite of tools. You get Calico with every purchase of Costar.

Costar	Estimation
Calico	Calibration, Tuning, Tweaking the model

By now, you know that Costar is used to model & estimate a software project.

Calico is used to calibrate the COCOMO equations to your local environment, based on projects you've already completed. You can do periodic Calico calibrations as you complete more projects, and feed the revised equations into Costar. You might even have multiple calibrations -- perhaps one for MIS applications that you do in Cobol, and another that you use to estimate your 4GL

projects. Most Costar users never use Calico, but we recommend it to everyone who's collected enough data.

Calico lets you change any of the thousands of numbers that control a COCOMO estimate, including:

- You can adjust the Hours per Person-Month.
- You can modify the default settings for Cost Drivers. You might set the PCAP (Programmer Capability) to High as your company's default.
- You can change the Effort Multipliers that are lurking behind each Cost Driver.
- You can define your own Cost Drivers.
- You can change the estimating equations -- but you'd generally be better off using Calico.
- You can change the Effort Distribution. If your organization spends more time coding, and less time testing, you can describe that. You can change the Schedule Distribution to match that.
- You can change the Activity Distribution.
- You can define your own Labor Classes (e.g. Programmers, Analysts).
- You can set your currency symbol.
- You can set the default Scale Drivers.
- You can change the Function Point weighting.
- You can set the default calculations to either Intermediate or Detailed.
- ... and lots more...

Back to Top

---

13. Should I use the Detailed calculations?

COCOMO II is defined using intermediate calculations, but the COCOMO 81 models let you choose between the Intermediate mode and the Detailed mode.

The two modes are equally fast, but the Detailed mode use phase-dependent effort multipliers for the cost drivers. For example, The Programmer Capability (PCAP) cost driver set to Very High in COCOMO 81 has an effort multiplier of 1.00 for the Product Design phase, but an effort multiplier of 0.65 for the Code & Unit Test phase. That means that the quality of the programmers doesn't have an impact on the design, but will save you a lot of effort in the programming phases.

Back to Top

---

14. How did Costar get this answer?

If you try to do the COCOMO calculations by hand, but can't match the Costar results, consider these ideas:

- You may be doing the simpler Intermediate calculations. You can tell Costar to do them too, so that your can compare answers.
- In some of the later COCOMO models, a cost driver setting of Nominal may not correspond to an Effort Multiplier of 1.00, so you may have computed the effort incorrectly. For example, a Nominal setting for LEXP in the ADA_87 model corresponds to an Effort Multiplier of 1.04 for the Intermediate model.
- You may be using the APM_88 model, but forgot to interpolate based on the "Ada Process" factor (also known as "sigma"). ACAP, PCAP, the Schedule Distribution Table, and the Effort Distribution Table all require interpolation in the APM_88 model.
- If your estimate is using Incremental Development, be sure to look at the Effort & Breakage report. Breakage Effort isn't assigned to a specific component.

Call us if you can't resolve the problem.

Back to Top

---

15. How should I model reuse of code?

If you are reusing code that already exists, you should use the COCOMO adaptation equations.

If you're writing code that will be reused, you should use the RUSE cost driver (available in all of the models except COCOMO_85).

The idea behind the RUSE cost driver is that it is more expensive to develop code that is designed and documented to be reused -- it's harder than simply making a special purpose version of the code. The payoff only comes when you use the code a second and third time. Which explains why reuse of code is the exception rather than the rule -- nobody has an incentive to make life easier for the next generation of developers. Usually, we're too busy trying to ship a product.

Back to Top

---

16. Which Costar model should I use?

We include 13 different models with Costar 7.0. Of course, you can use Calico to make your own versions.

If you're new to COCOMO and Costar, you should probably use one of these Post-Architecture models for your estimating:

- COCOMO_II.2000_Waterfall
- COCOMO_II.2000_MBASE

Select the model that most nearly matches your development environment. The Waterfall model has a traditional set of software development phases; the MBASE model is comparable to the Rational Unified Process.

The COCOMO II Early Design models are intended for use when very little is known about the project you're estimating. Instead of the full complement of 17 cost drivers available in the Post-Architecture models, the Early Design models have 7 composite cost drivers. Otherwise, the models are the same.

There are also four COCOMO II.1997 models built into Costar 7.0. Use these models only for compatibility with existing estimates -- otherwise, you'd be better off using a COCOMO II.2000 model.

You may see references in the literature to COCOMO II.1998 or COCOMO II.1999. COCOMO II.1998 was renamed to COCOMO II.1999 and then to COCOMO II.2000, so all of those models are identical.

These five models built into Costar are variations on the original COCOMO 81 model;

- REVIC
- APM 88
- Ada 87
- COCOMO 87
- COCOMO 85

COCOMO_85

This is almost identical to the COCOMO defined in ***Software Engineering Economics***. It simply has an extra setting for the TURN and TOOL cost drivers.

If you want to use the most standard version of traditional COCOMO, use this model.

COCOMO_87

This database uses the COCOMO 81 equations, but has 4 new Cost Drivers added: VMVH, VMVT, SECU (Security), and RUSE (Reuse). Two of the standard cost drivers have been updated -- SCED and TOOL.

Ada_87

This is like the COCOMO_87 model, except that we assume that you're using Ada as a programming Language. The RELY cost driver has changed to reflect that it's easier to create reliable software in Ada. The CPLX cost driver has changed to reflect that it's easier to develop complex software in Ada. The LEXP cost driver penalizes you for being an inexperienced Ada programmer.

Use this model if you're developing software in the traditional fashion, but you're writing in Ada.

APM_88

This model has a different form for the equations, new tables, new cost driver values, etc. It assume that you've adopted a new way of developing software -- the Ada Process Model (APM).

Use this model after you've read the relevant papers. This is considered an experimental model, so you should use it in parallel with one of the more conventional models.

It's interesting to compare this experimental model to the form of the COCOMO II model. The major feature introduced with the Ada Process Model, and carried forward into COCOMO II, is the idea that there are a set of Scale Drivers that modify the exponent in the estimating equation. The original COCOMO models only have a single factor, the Development Mode, that changes the exponent.

Back to Top

---

17. I've got old WICOMO estimates. Can you help me convert them?

Sure. Version 1.00 of Costar could read the old WICOMO estimates, but there hasn't been much demand for that lately, so the latest Costar doesn't support that feature.

We can help customers convert the WICOMO estimates into Costar estimates.

Back to Top

---

18. Can I compare one Costar estimate to another?

Yes. You can have any number of Costar estimates in memory at the same time, and you can use the Comparison Report to compare up to 3 of them side-by-side.

We want you to develop many version of your estimate. You should feel free to try different experiments as you refine the model of your software project. Here are the sort of experiments you might try:

- Does it make sense to spend money on new hardware for the development team (i.e. to reduce TURN, the turn around time cost driver)?
- Does it make sense to spend money on new software development tools? That might let you save time by letting you change the TOOL cost driver to a more advantageous setting.
- Is it cheaper to use our most capable (and most costly) people on this project? Set ACAP and PCAP to high ratings, and adjust the RQCOST, PDCOST, and DDCOST parameters.
- How much time would we save if we used 2 overlapping increments, instead of trying to develop the project as a single monolithic chunk?

Back to Top

---

19. How do I get started?

Read the Costar Overview , the COCOMO Overview, and the page about Advanced COCOMO.

Next call us, or send email requesting a demo disk. Better yet, download the demo!

NEW Download a demo of Costar 7.0 for Windows. NEW

If you're ready to place an order, check the Price List, and then contact us.

Back to Top

---

20. When will the Linux and Mac versions of Costar be released?

There's no release currently planned. Send email to let us know what you'd like to see.

Back to Top

---

21. Is COCOMO the best software estimation model?

We like it, and it is the most popular model.

But you don't need to take our word for it -- COCOMO is so good at estimating, that one of our major *competitors* (a Fortune 100 company) has a worldwide license for Costar and uses it extensively for their internal estimation projects. What better endorsement?

To do a good job estimating, you should use 2 or more independent techniques. One of them must be COCOMO. Even if you don't buy our product, you should use COCOMO for your estimating. You can do small estimates by hand if you need to, or get a simple, free, version from USC, but you ought to use COCOMO.

You have a lot of choices for your second technique. We have a number of competitors -- call us and we can recommend our favorites (most of them will suggest that you use 2 or more techniques, too). The best choice would be an estimating approach that differs significantly from the COCOMO technique, so that you examine your project from more than one point of view.

It's easy for us to recommend that you talk to our competitors. Costar is much less expensive then their products, so we think you'll be back to talk to us. Ask to see their Price List.

"Expert Judgment" *is* an acceptable technique. If you have experienced estimators that can give you good results, use them.

Back to Top

---

22. Can Excel and Costar trade data?

All of the Costar reports can export their data to Excel. They can even do it in real time, so that as you change parameters in Costar, all of your related spreadsheets change. So, if none of the 18 built-in reports & graphs show what you want to see, you can use Excel as a fancy report writer/graphing tool.

Importing data to Costar is a bit more tricky. Click here to download an Excel spreadsheet that does just that. It has a macro that collects data from the spreadsheet, executes Costar (sending the data), and harvests the results from a Costar report, and places the Costar results into the spreadsheet. The macro assumes that Costar is installed in the "Costar 7" directory or "Costar 7 Demo" directory.

You can embellish this simple macro into something that is more useful. Contact us if you need more details about the Costar command language.

By the way, when you read the spreadsheet into Excel it *will* warn you about the presence of a macro -- that's normal in this case.

Software project managers need a tool to help plan and control their projects.

For starters, a software cost estimation tool should offer:

Accurate estimates	Within 20% of actuals, 70% of the time
Automatic recalculation	For fast operation; can be toggled on and off
Comprehensive estimates	Estimates for Requirements, Design, Coding, Testing, Maintenance, and Adaptation
A track record	Used by thousands of software project managers

Some of the better software cost estimation tools may also provide:

What-if analyses	Make trade-offs & perform sensitivity analyses
Extensibility	User definable cost drivers
Adaptability	Can be calibrated to your environment
Function Points	An optional software sizing method

There is one software cost estimation tool that gives you ALL of the above PLUS:

Easy to learn interface	Context sensitive help, pull down menus
COCOMO II.2000	The latest and most accurate COCOMO calibration
An established model	Based on COCOMO, not a proprietary model, so all details are published
Schedule Constraint Optimization	A GUI tool that lets you make schedule/effort trade-offs -- you can constrain a project to a certain duration or cost
Support for Ada COCOMO	Select either COCOMO II, COCOMO 81, or Ada COCOMO estimates

Support for Incremental COCOMO	Model your Incremental Development or Prototyping process
REVIC	Our REVIC implementation uses the same equations, cost drivers, and phase distribution as the popular DOS program
21 different reports & graphs	Including schedule, staffing, cost, summary, activities, and milestones -- each report can be sent to the screen, printer, or a file
Export to Excel	Any of the reports can be exported; you can even due live updates of Excel spreadsheets
Work Breakdown Structures	Any number of levels in product hierarchy (system, subsystem, module, etc.) -- You can describe your project's structure in as much detail as you choose

Which tool? COSTAR. We know you need professional tools, AND:

Complete estimation package	• Includes Barry Boehm's ***Software Engineering Economics*** (767 pages). • Includes ***Software Cost Estimation with COCOMO II*** (502 pages).
Volume discounts	Site license available, Corporate license available
Support after the sale	One year of free phone support

12 billion reasons to use Costar:

5	Operating Systems (Win 95, Win 98, Win NT, Win 2000, Win XP)
13	COCOMO estimating models built in
17	Years of enhancements & improvements
1,000,000	Licensed users (almost 2 million now!)
18	Reports
3	Graphs (you can use live links to Excel to make your own)
1,269	Textbook pages describing COCOMO
32	Cost Drivers (and you can add your own)

1	Web site for information & support
(603) 672-0987	Phone number for technical support
(603) 672-3460	Fax number for technical support
12,074,445,805	Total

The Product

- **The Costar 7.0 estimation program**
- **The Calico 7.0 calibration program**
- **Free, unlimited, phone support for 1 year.**
- **Free upgrades for 6 months.**
- **1 complimentary copy of Dr. Barry Boehm's *Software Cost Estimation with COCOMO II* included with each order.**
- **1 complimentary copy of Dr. Barry Boehm's *Software Engineering Economics* included with each order.**

Pricing

Single Copy License	**$1,900**
Site License	**$5,000**
Corporate License **Special limited time offer:** **Free training included**	**$25,000**

Call us at (603) 672-0987 to get a quote on upgrading your current copy of Costar to Costar 7.0 -- we offer discounts of up to 80% for our current customers!

Each price is a one time charge, NOT an annual fee.

By the way, as you shop for a software cost estimation tool, please notice that **none of our competitors post a Price List**. There's a reason for that. Most of the other tools cost several thousand dollars **per seat, per year**.

The Costar Guided Tour starts here...

The guided tour will lead you through a simple application of our COCOMO II software estimation tool.

Guided Tour -- Costar Wizard (13 page tutorial)...

Guided Tour (11 page tutorial)...

The tutorials covers just a fraction of Costar's capabilities. The following links show most of Costar's features and reports, so that you can get a better idea of what Costar can do for you.

The Main Costar Window (1 page)

Menus (7 pages)

Estimate Toolbar (1 page)

Component Toolbar (1 page)

Reports Toolbar (1 page)

Notebook Tabs (12 pages)

Reports (13 pages)

Help Table of Contents (1 page)

Worksheets (8 pages)