The structure of empirical estimation models is a formula, derived from data collected from past software projects, that uses software size to estimate effort. Size, itself, is an estimate, described as either line of code (LOC) or function points (FP).

No estimation model is appropriate for all development environments, development processes, or application types. Models must be customized (values in the formula must be altered) so that results from the model agree with the data from the particular environment.

The typical formula of estimation models is: E = a + b(S)c

Where; E represents effort, in person-months,

S is the size of the software development, in LOC or FP, and, a, b, and c are values derived from data.

The relationship seen between development effort and software size is generally: E S the amount of effort accelerates as size increase<http://athena.ecs.csus.edu/~buckley/CSc231_files/COCOMO.pdf>s, i.e., the value c in the typical formula above is greater than 1. COCOMO:

The original COCOMO model was a set of models; 3 development modes (organic, semi-detached, and embedded) and 3 levels (basic, intermediate, and advanced).

**Basic**-predicted software size (lines of code) was used to estimate development effort.

**Intermediate-**predicted software size (lines of code), plus a set of 15 subjectively assessed 'cost drivers' was used to estimate development effort.

**Advanced** - on top of the intermediate model, the advanced model allows phase-based cost driver adjustments and some adjustments at the module, component, and system level.

COCOMO development modes:

**Organic -** small relatively small, simple software projects in which small teams with good application experience work to a set of flexible requirements.

**Embedded** - the software project has tight software, hardware and operational constraints.

**Semi-detached -** an intermediate (in size and complexity) software project in which teams with Where:

**E** represents effort in person-months,

**S** is the size of the software development in KLOC and,

**a and b are values** dependent on the development mode,

development mode:

organic a = 2.4 b = 1.05

semi-detached a = 3.0 b = 1.12

embedded a = 3.6 b = 1.20d

experience levels must meet a mix of rigid and less than rigid requirements.

1 The intermediate and advanced COCOMO models incorporate 15 'cost drivers'.

These 'drivers'

multiply the effort derived for the basic COCOMO model. The importance of each driver is assessed and the corresponding value multiplied into the COCOMO equation, which becomes:

**E = a(S)b x product(cost drivers)**

As an example of how the intermediate COCOMO model works, the following is a calculation of

the estimated effort for a semi-detached project of 56 KLOC. The cost drivers are set as follows:

Product cost drivers (from the table) set high = 1.15 x 1.08 x 1.15 = 1.43

Computer cost drivers (from the table) set nominal = 1.00

Personnel cost drivers (from the table) set low = 1.19 x 1.13 x 1.17 x 1.10 x 1.07 = 1.85

Project cost drivers (from the table) set high = 0.91 x 0.91 x 1.04 = 0.86

hence, product(cost drivers) = 1.43 x 1.00 x 1.85 x 0.86 = 2.28

for a semi-detached project of 56KLOC: a = 3.0 b = 1.12 S = 56

E = a(S)b x product(cost drivers)

E = 3.0 x (56)1.12 x 2.28

E = 3.0 x 90.78 x 2.28

E = 620.94 person-months

**Description, and Table of values, for COCOMO Cost Drivers:**

COST DRIVER DESCRIPTION RELY

Required software reliability

DATA Database size

CPLX Product complexity

TIME Execution time constraints

STOR Main storage constraints

VIRT Virtual machine volatility - degree to which the operating system changes

TURN Computer turn around time

ACAP Analyst capability

AEXP Application experience

PCAP Programmer capability

VEXP Virtual machine (i.e. operating system) experience

LEXP Programming language experience

MODP Use of modern programming practices

TOOL Use of software tools

SCED Required development schedule

COCOMO - **COST DRIVERS RATING**

**COST V.LOW**

**DRIVER**

|  |  |
| --- | --- |
| LOW NOMINAL HIGH V.HIGH EX. HIGH  (PRODUCT) | RELY 0.75 0.88 1.00 1.15 1.40 |
| DATA | 0.94 1.00 1.08 1.16 . |
| CMPLX | 0.70 0.85 1.00 1.15 1.30 1.65 |
| CMPUTE | . 1.00 1.11 1.30 1.66  .  .. |
| CMPLX COMPUTER TIME | 1.00 1.11 1.30 1.66 |
| STOR | 1.00 1.06 1.21 1.56 |
| VERT | . 0.87 1.00 1.15 1.30 |
| TURN | 0.87 1.00 1.07 1.15 . |
| .. PCAP.  (PROJECT)  MODP..  TOOL  LEXP  SCED | . 1.46 1.19 1.00 0.86 0.71 .    1.14 1.07 1.00 0.95 . .  1.24 1.10 1.00 0.91 0.82 .  1.24 1.10 1.00 0.91 0.83 .  1.23 1.08 1.00 1.04 1.10 . |
| VEXP | 1.21 1.10 1.00 0.90 . .  1.42 1.17 1.00 0.86 0.70 |

**List down all the requirements of the project and do the** [**estimations**](https://www.pmi.org/learning/featured-topics/project-estimating) **for them using Planning Poker, Bucket System, Fibonacci series etc. All the team members should agree upon the estimations done for the listed requirements after clear analysis and understanding of the user stories. Estimations are done based on the features to be implemented in a user story.**

Given below are the 3 main levels of Agile Estimation.

#1) Project or Proposal level is the one that uses Quick Function Point Analysis during the initial phases of the project development.

#2) Release Level includes assigning the story points to the user stories that can help in defining the order of the user stories based on the priority and can also help in deciding which stories can be taken in the current release and which can be taken later.

#3) Sprint Level is the one where the user stories are broken into the tasks and estimated hours are assigned to the tasks according to their complexity. Here, we also define the person responsible for the task along with the status of the tasks.

This information can be later used to calculate the budget for the Agile project. Calculation of Budget is crucial to make sure that the project does not go over the budget due to the pre and post iteration tasks or some other reasons.

Story Points Estimations In Agile

Story Points estimations is a comparative analysis to roughly estimate the product backlog items with relative sizing. The team members for estimating user stories include Product Owner, Scrum Master, Developers, Testers, and Stakeholders.

Given below are few steps to reach the final decision of relative sizing:

#1) Analyze all user stories and identify the base or reference story. It is important for doing relative sizing. This story can be chosen from the current product backlog or the one, that we have done earlier. This story should be chosen as the reference story upon agreement of all members.

#2) Pick another story from the current Product Backlog and the team members are free to discuss any questions or doubts with the Product Owner while understanding the requirements of the story. Product Owner is responsible for clarifying all their queries and doubts.

#3) Make a list of the things to be taken care of while implementing the user story. These can be done by writing notes in the notes section of the tool or by adding bullet points on the story card. This is mostly done by the Scrum Master.

**#5)** Do relative sizing for the story selected. If it requires the same amount of work and effort, then assign it the same no. of points, as assigned to the reference story. If it requires more effort, assign it some higher value. If it requires less effort, assign it some lower value.

**#6)** Reach a consensus with all the participants to finalize the relative size for the selected user story as per the definition of done.

**#7)** After relative sizing of all the product backlog items has been done, ensure that if all the user stories with the same no. of points assigned to them, require the same effort and size to be consistent.

**There are mainly 7 Agile Project Estimation Techniques:**

#### #1) Planning Poker

#### 2. Dot planning

3. T-shirt

4.The bucket system

5. The affinity system