Logic of Software Estimation and Tools

Anandi Hira

Roundtable Discussion
November 9, 2022

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Document Markings

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM22-0936

Agenda

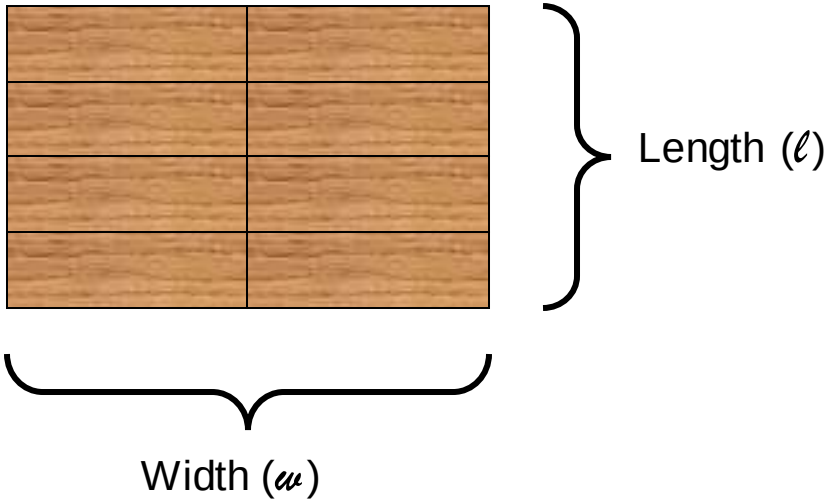
1. Foundational Concepts
2. Software Cost Estimation Tools
3. Inherent Biases and Possible Solutions
4. Software Size Metrics

Logic of Software Estimation and Tools

Foundational Concepts

How Do You Estimate a Flooring Job?

Area (a) = $\ell \times w$



Flooring job cost estimating model:
a × (floor unit price + scrap % × floor unit price + labor unit price + additional materials)

- The major size metric is a (a major driver of the cost).
- The floor unit price is a cost driver, which changes the cost based on specific user choices (e.g., expensive vs. affordable flooring choice).

How to Estimate a Software Job?

```
#include <iostream>
using namespace std;

int main() {
    int i = 0;
    while (i < 5) {
        cout << i << "\n";
        i++;
    }
    return 0;
}
```

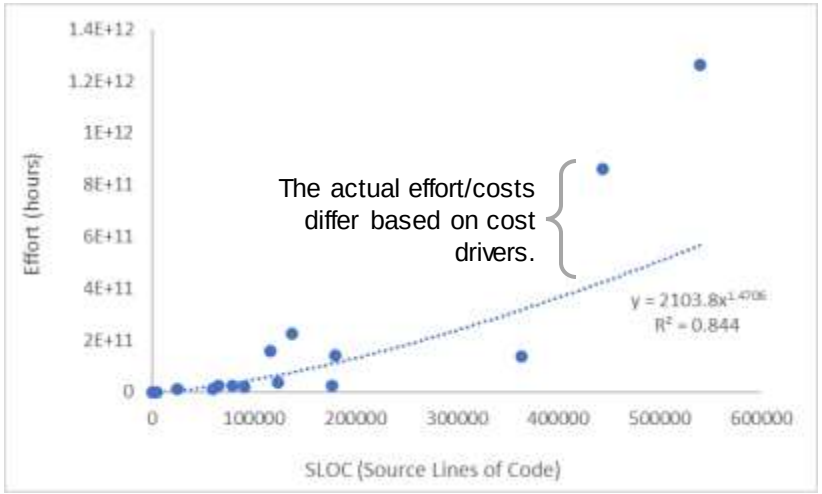
Using the previous example, we must identify

1. Size metric
2. Cost drivers
3. Relationship among size, drivers, and effort/cost

Size Metric

- The direct output is code.
- Count the source lines of code (SLOC).

Why Do We Need Size *and* Cost Drivers?

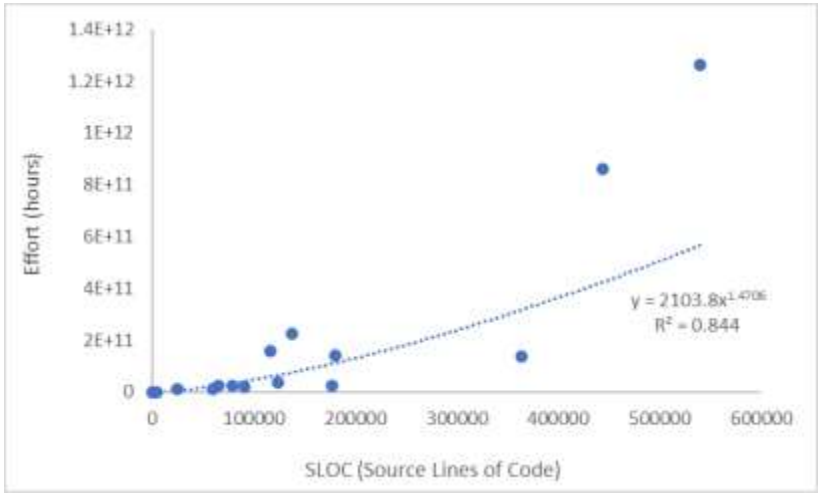


While size provides an average effort/cost, the actual effort/cost must be adjusted to consider positive and negative factors.

Categories of cost drivers for software include

- Product (e.g., reliability requirements, storage constraints)
- Personnel (e.g., experience, processes used)
- Environmental (e.g., team co-location, tools usage)

Software Relationship Between Effort/Cost, Size, and Drivers



Generally, software cost/effort estimation models use a power equation:

- The equation is $a \times size^b \times$ cost drivers.
- Effort/cost grows at nonlinear rate.
- Cost drivers have a multiplicative effect on effort/cost.

Characteristics of Good Estimates

Accurate

- Use models based on high-quality, representative data
- Understand model before use
- Adjusted for inflation, cost drivers/factors, uncertainty and bias

Credible

- Account for uncertainty and bias
- Include sensitivity analysis and cross-checks with independent estimates

Reproducible

- Are well documented, especially ground rules and assumptions
- Use a mathematical model (to yield the same results with inputs)

Comprehensive

- Are based on a complete technical specification
- Account for all program elements and components

Software Cost Estimation Tools

COCOMO II: Model Summary

$$PM = A \times Size^{(B + C \times (\sum SF))} \times \prod EM$$

The exponent ranges from 0.9 to 1.2, with 1.0997 as the default.

PM—Software development effort (in person-months)

Size—Size in thousand SLOC (KSLOC)

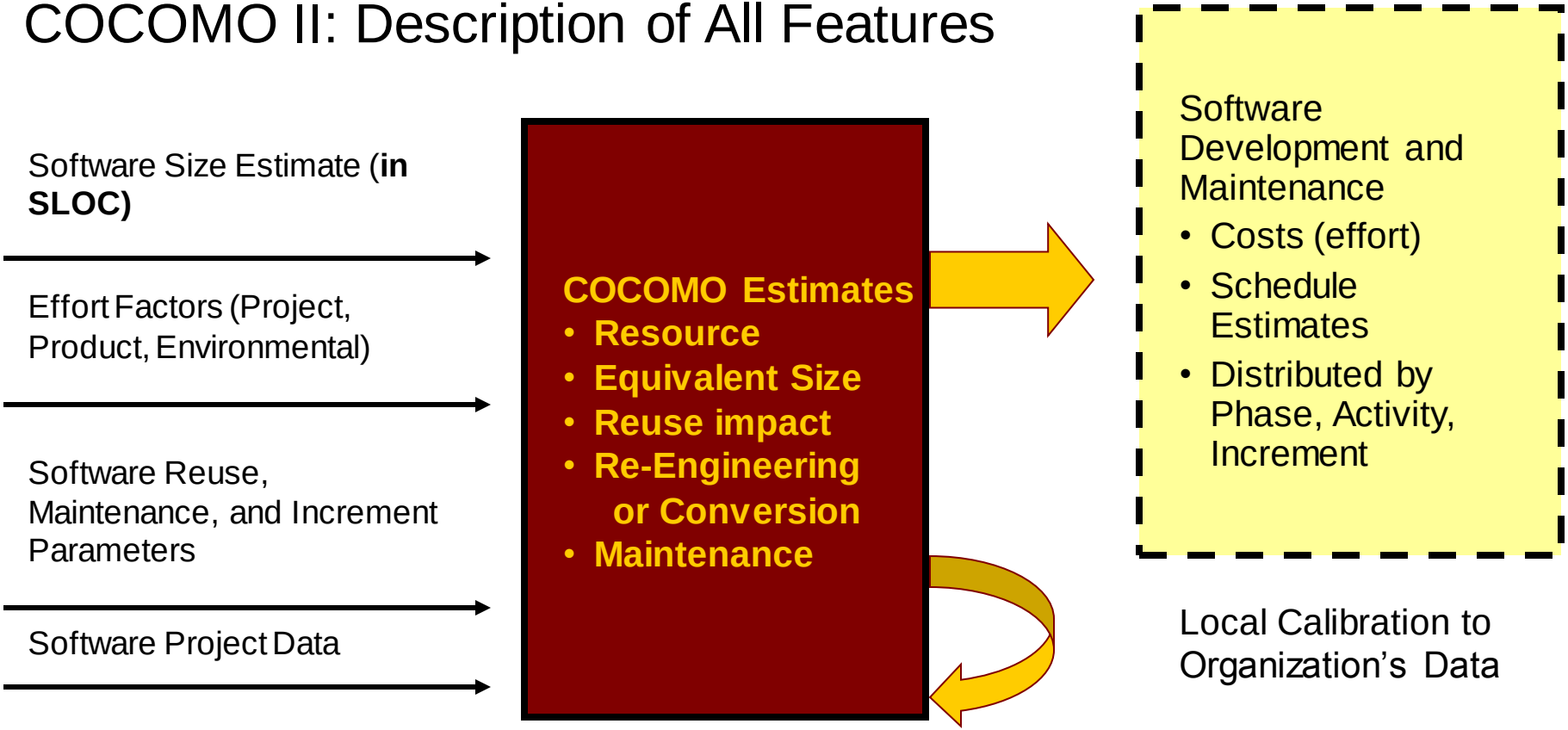
A—Calibrated productivity constant (KSLOC/PM)

B, C—Calibrated exponent constants

SF—Scale factors (have exponential effect)

EM—Effort multipliers (have multiplicative effect)

COCOMO II: Description of All Features



COCOMO II is an open and free model.

SEER-SEM by Galorath

$$\text{Effort} = Lx \times (\text{AdjFactor} \times \text{FSM})^{\left(\frac{\text{Entropy}}{1.2}\right)}$$

Lx—Effort units

AdjFactor—Product of complexity, count phase, platform and application adjustments (factors/drivers)

Entropy—Ranges from 1.04 to 1.2, depending on the type of software

- Includes thousands of data points from Department of Defense (DoD) and commercial products
- Allows several types of software size metrics
- Provides risk information

TruePlanning by Unison Software

$$\text{Effort} = \text{Size} \times \text{Baseline Productivity} \times \text{Productivity Adjustments}$$

Baseline Productivity—Varies by activity and size metric used, using existing data or research results

Productivity Adjustments—Numerical effects of cost drivers/factors on productivity

- Data across various domains (e.g., business systems, military, avionics, flight and space software, and commercial software)
- Allows several types of software size metrics

Inherent Biases and Possible Solutions

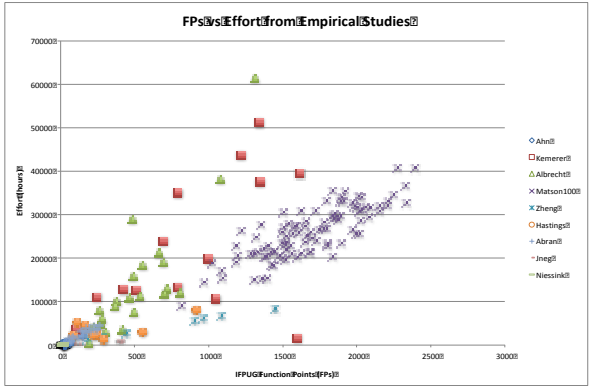
Data Varies Across Teams and Organizations

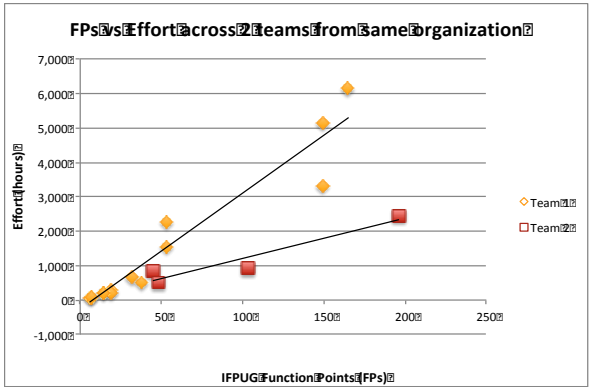
These graphs show how different trends between size and effort behave:

- Top: across different organizations
- Bottom: across different teams in one organization

How do you build a generalizable model?

- Use cost drivers that normalize the differences.
- Calibrate the existing model to the organization/team's data.





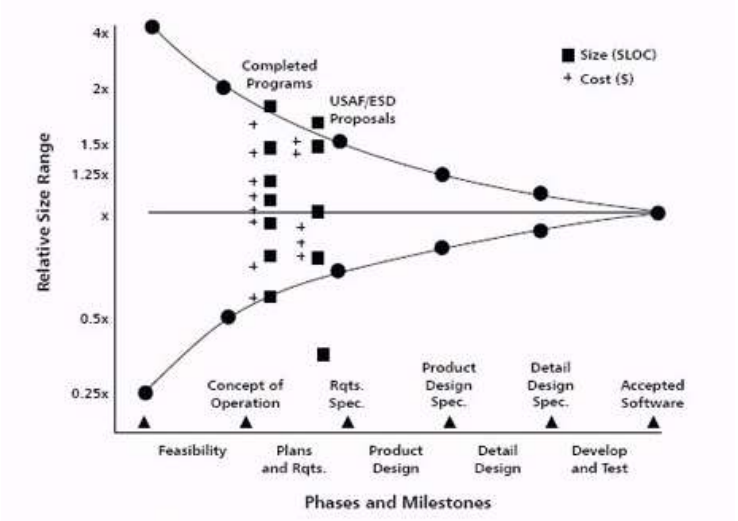
Requirements Uncertainty and Volatility

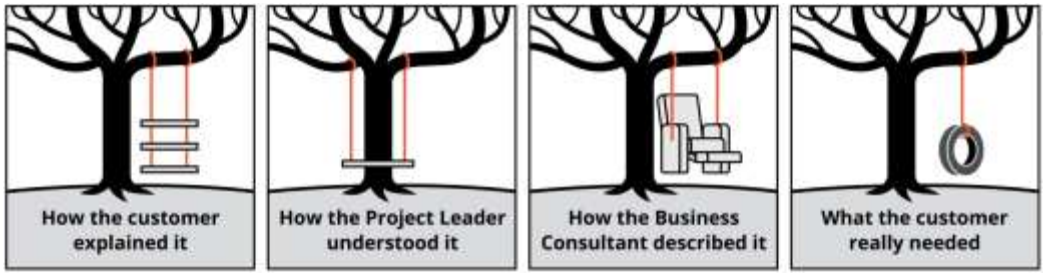
Estimates are based on estimated size (i.e., a representation of what will be developed).

As specifications change, so does the size. Estimated size has low accuracy, especially early in the lifecycle.

How to account for uncertainty/volatility?

- Incremental development
- Uncertainty and risk analysis
- Multiplying size with a volatility factor
- Updating estimates as specifications change





New Project Not Represented in the Model

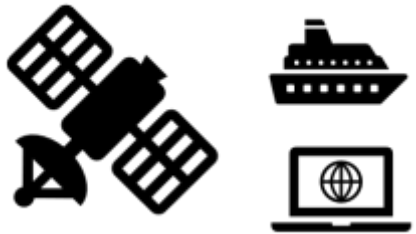
The data used to build models determines what the model can estimate with some amount of certainty.

How to Estimate

- Use cost drivers that normalize the differences.
- Calibrate the existing model to data on similar types of projects.
- Analyze uncertainty and risk.

Types of Projects Represented in the Model

What Must Be Estimated



Logic of Software Estimation and Tools

Software Size Metrics

Challenges Estimating SLOC

- Estimating SLOC is a subjective process, equivalent to guessing how much candy is in a jar.
- Despite having estimating experience, estimating SLOC is difficult.
- Adding different types of components, reuse, services, etc. further complicates the process.

Functional Size Metrics

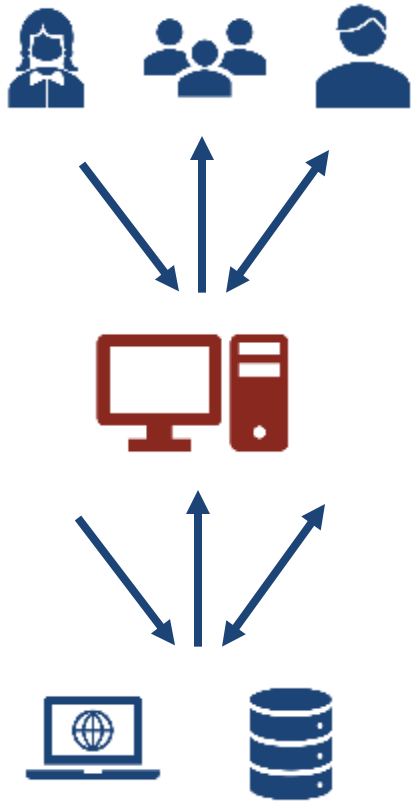
They are based on the software's functionality.

Pros

- Objective rules for size measurement
- Decent traceability from requirements to size

Cons

- Tedious to calculate
- Requires understanding architecture/design of software
- Learning and application curve



- Inputs
- Outputs
- Queries

- Store to and Access from Internal Storage

And

- Access External Data

Story Points

They are based on user stories (requirements). Each team member estimates the size and then discusses it to reach agreement.

Pros

- Easier to estimate with high-level information (user stories)
- Allows team to discuss requirements
- Decent traceability from requirements to size

Cons

- Subjective: sizes not consistent across teams

Software Sizes by Granularity

Lifecycle Phases	Feasibility	Plans and Requirements		Product Design		Detail Design	Develop and Test
Requirements Levels	Summary Goals		User Goals		Sub-functions		
Sizing Aids	Product Vision, Analogies	Operational Concept, Context Diagram	Specification, Feature List	Architecture, Top-Level Design		Detailed Design	Code
Size Metrics	• Key Features • Story Points (2004)	• User Roles • Use Cases • Use Case Points (1993)	• Screens • Reports • Files • Object Points (1992)	• IFPUG Function Points (FPs) (1979) • COSMIC Function Points (CFPs) (1999)	• Components	• Objects • Predictive Object Points (1997)	• Source Lines of Code (1960s)

Summary

Summary

Foundational Concepts

- Power equation, size, and cost drivers

Software Cost Estimation Tools

- COCOMO II, SEER-SEM, Unison TruePlanning

Inherent Biases and Possible Solutions

- Data variation across teams and organizations
- Requirements uncertainty and volatility
- New project not represented in the model

Software Size Metrics

- SLOC, functional size, story points, sizes vs. granularity