CS445 / ECE451 / CS645 / SE463

Software Requirements Specification & Analysis

Estimating Software Size

Data to Collect

In order to compute software estimates based on historical data, we need to be collecting data about our projects.

- Project size
  - lines of code (LOC)
  - requirements
    - e.g., function points, scenarios, stories, UI screens
- Effort (staff months)
- Time (calendar days, weeks, months)

Collecting data on size and effort every 1 to 2 weeks can provide valuable insight into our project's dynamics.

Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006.

#LOC

Space Shuttle (primary flight software)	400,000
LibreOffice	9.1 million
Android (Sept. 2015)	12 million
Boeing 787 Dreamliner	14 million
Microsoft Office	~45 million
Ford F150 Truck (2016)	150 million
Debian 5.0 (all software in package)	324 million
Healthcare.gov	500 million
All of Google's Internet Services - search, gmail, maps, etc. (Sept. 2015)	2 billion

https://docs.google.com/spreadsheets/d/1s9u0uprmuJvwR2fkRqxJ4W5Wfomimmk9pwGTK4Dn_UI/edit#gid=5

How to Count LOC

- Do you count all code or only code that's included in the released software?
- How do you count code that's reused from previous versions?
- How do you count open-source code or third-party library code?
- Do you count blank lines and comments
- Do you count class interfaces?
- Do you count data declarations?
- How do you one logical line of code that is broken across multiple lines for the sake of readability?

How to Measure Effort

- How many hours per day do you count? Standard 8 hours or actual hours applied to the specific project?
- Do you count unpaid overtime?
- Do you count holidays, vacation, and training?
- Do you make allowances for all-company meetings?
- What kinds of effort do you count? Testing? Documentation? Requirements? Design? Research?
- How do you count time that's divided across multiple projects?
- How do you count time spent supporting previous releases of the same software?
- How do you count time spent supporting sales calls, trade shows, and so on?
- How do you count travel time?

Estimates Based on Past Data

Compute estimates of new development tasks based on the actual work to complete past tasks.

- Project data - data generated earlier in the same project that's being estimated
- Historical data - data from the organization that will conduct the project being estimated
- Industry data - data from other organizations that develop the same basic kind of software as the software that's being estimated

Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006.

#1 Estimation by Analogy

Compute a size estimate based on a piece-by-piece count of analogous elements, and the past sizes of those elements.

Subsystem	Old Project (Size)	New Project (Est. Size)	Multiplier	Old Project (LOC)	New Project (LOC)
Use Cases	10 use cases	14 use cases	1.4	5,000	7000
User Interface	14 screens	16 screens	1.1	14,000	15,400
Graphs	10 graphs	16 graphs	1.6	9,000	25,600
Reports	3 reports	5 reports	1.7	3,000	5,100
Bus					16,500
TOT					**69,600**

Expert estimators: decomposing the problem into pieces, estimating each piece by analogy (calibrated with historical data), arrive at estimates that have a standard deviation of roughly 7%

Historical data
Estimates
Computed values

Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006. (Tables 11.1, 11.2)

#2 Function Point Analysis

Goal: More precise estimates based on product complexities (known at requirements)

We break the problem down into three parts:

1. Estimate the number function points from the requirements
2. Estimate code size from function points
3. Estimate resources required (time, personnel, money) from code size

Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006.

Function Points

A function point is a size metric of functionality. It relates to the requirements of the software under development.

The number of function points in a system is based on the number and complexity of the following

- **External Inputs** – screens, forms, dialog boxes, messages
- **External Outputs** – screens, reports, graphs, messages
- **External Queries** – responses to input queries that do not change system data
- **Internal Logical Files** – major internal data stores (end-user data, control information)
- **External Interfaces / APIs** – files or APIs controlled by adjacent systems

Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006.

Function Point Counting

A function point count is computed by adding up inputs and outputs, weighted by complexity multipliers.

Program Characteristic	Low Complexity	Medium Complexity	High Complexity
External Inputs	___ x 3	___ x 4	___ x 6
External Outputs	___ x 4	___ x 5	___ x 7
External Queries	___ x 3	___ x 4	___ x 6
Internal Files	___ x 7	___ x 10	___ x 15
External Interfaces / APIs	___ x 5	___ x 7	___ x 10

Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006. (Table 18.1)

Function Point Counting

A function point count is computed by adding up inputs and outputs, weighted by complexity multipliers.

Program Characteristic	Low Complexity	Medium Complexity	High Complexity
External Inputs	6 x 3	2 x 4	3 x 6
External Queries			
Internal Files	0 x 7	2 x 10	3 x 15
External Interfaces / APIs	2 x 5	0 x 7	7 x 10

Studies have shown that certified function-point counters (www.ifpug.org) usually produce function-point counts that are within 10% of each other.

= 284 function points

Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006. (Table 18.2)

Estimating Code Size from FPs

Language	Average	Median	Low	High
ASP	51	54	15	69
C	97	99	39	333
C++	50	53	25	80
C#	54	59	29	70
Excel	209	191	131	315
HTML	34	40	14	48
J2EE	46	49	15	67
Java	53	53	14	134
JavaScript	47	53	31	63
.NET	57	60	53	60
Oracle	37	40	17	60
SAS	38	37	22	55
SQL	21	21	13	37

http://www.qsm.com/resources/function-point-languages-table

#3 Structured Expert Judgment

Use expert judgment as a last resort, or for a second or third estimation.

- People doing the work will create the most accurate estimates
- It's best to decompose estimation – and estimate tasks that require no more than 2 days of effort
- It's best to estimate a range (best case vs. worst case) of sizes
  - Single-point estimates tend to be much closer to best-case estimates
- Program Evaluation and Review Technique (PERT) Formula

ExpectedCase = [BestCase + (4*MostLikelyCase) + WorstCase] /6
ExpectedCase = [BestCase + (3*MostLikelyCase) + (2*WorstCase)] /6

Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006.

#4 Commercial Estimation Tools

Commercial tools embed computationally intensive estimation methods that cannot be done easily by hand or with a calculator.

- Complex effort estimations that account for diseconomies of scale
- Simulating project outcomes (with respect to probability distributions for variability in system size, productivity)
- Probability analysis of estimates

Tools tend to have large databases of industry-average data, but they can be calibrated with your own historical data.

Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006.

Summary of Size Estimation Tech.

Technique	Kinds of Sizes that can be Estimated
Analogy	features, function points, screens, GUI components, LOC
Estimation Tools	function points, LOC
Function Point Analysis	function points, LOC
Structured Judgment	features, user stories, requirements, use cases, function points, Web pages, GUI components, database tables, interface definitions,
GUI Elements	function points, LOC
Standard Components	function points, LOC
Story Points	story points, LOC
Wideband Delphi	features, user stories, requirements, use cases, function points, Web pages, GUI components, database tables, interface definitions, classes, functions/subroutines, LOC

It's best to create Best-Case, Worst-Case, AND Expected-Case estimates.

Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006. (Table 18.5)

References

McConnell, S., *Software Estimation: Demystifying the Black Art,* 2006.