Software Engineering

Richard Bartels

University of Waterloo
Faculty of Mathematics
Department of Computer Science

November 20, 1996

Metrics and Estimation

Richard Bartels

University of Waterloo
Faculty of Mathematics
Department of Computer Science

November 20, 1996

Outline

- What Are Software Metrics?
- Software Productivity
- Estimation Models
  - COCOMO

Reference:

This is material from Chapter 29 of Sommerville and from `http://www.spr.com/library/funcmet.htm` (by Capers Jones, Chairman, Software Productivity Research, Inc.)

Software Metrics

Attempts to answer one or both of the questions:

- How big is it?
- How good is it?

Which divides into:

- Before-delivery metrics
  - Estimate size/complexity
  - Use for planning costs/resources/time
- After-delivery metrics
  - Estimate quality
  - Use to review the company's processes and practices

Uses and Categories

Uses for metrics

- Project estimation and project monitoring
- Evaluation of work products
- Process improvement through failure analysis
- Experimental validation of best practices

Categories of metrics

- Productivity metrics
- Quality metrics
- Technical metrics

Each of these may involve

- Size-oriented metrics
- Function-oriented metrics
- Human-oriented metrics

Estimation

- Human effort
  - Skills needed
  - Numbers of people needed
  - Training and preparation needed
- Duration
  - Phases and milestones
  - Calendar time
- Cost
  - Wages
  - Tools
  - Infrastructure and support
- Opportunity and risk factors
  - Market opportunities
  - Estimate uncertainties
  - Contract terms
  - Requirements volatility
  - Financial health

What are the Products?

Standard Economic Definition of Productivity

Goods or services produced per unit of labor and expense

Goods or Services

Include

- requirements and specifications
- lines of code
- lines of code comment
- test cases
- test support
- planning documents
- user documentation
- training and technical support

Economics

Economic Rule

If a manufacturing process involves a substantial percentage of fixed costs, and the number of units manufactured is small, then the cost per unit must go up.

- contract software vs. market software

Paradox

Items such as specifications, test infrastructure, code documentation, version-control organization, etc. act as fixed costs. In higher-level languages their percentage increases relative to the lines of actual code. Thus, the more powerful the programming environment, the more cost per line of source code.

Estimation Methods

- Expert judgement
- Corporate experience
- Analogy
- What the market will bear
  (Use max of what customer can afford
  or what market research predicts)
- Parkinson's Law
  (Use max of available resources)
- Estimation model
  ("Algorithmic cost modelling")

We'll address models only (briefly)
Read Section 29.3 in the book!

Hypothetical Example (Capers Jones)

ACTIVITY	CASE A ASSEMBLER VERSION (10,000 LINES)	CASE B FORTRAN VERSION (3,000 LINES)	DIFFERENCE
Requirements	2 Months	2 Months	0
Design	3 Months	3 Months	0
Coding	10 Months	3 Months	- 7
Integration/Test	5 Months	3 Months	- 2
User Documentation	2 Months	2 Months	0
Management/Support	3 Months	2 Months	- 1
Total	25 Months	15 Months	- 10
Total Costs	$125,000	$75,000	- $50,000
Cost Per Source Line	$12.50	$25.00	+ $12.50
Lines Per Person Month	400	200	- 200

Approach

- Establish scope in advance
- Metrics are used as a basis for estimation
  - past measurements or predictions
- Project is broken into pieces for individual estimation
- Account for inherent complexity
  - Cyclomatic complexity
  - Function points
  - Feature points
- Normalize for language
  - Language tables
- Account for character of development environment
  - COCOMO

Estimates to be refined as the project progresses

Models to be tuned as more projects are accomplished
(Current models derive from management statistics from limited observations from selected corporations from certain projects taken as much as a decade ago)

Function Points

Due to A.J. Albrecht of IBM

Proposed in the late 1970's

Weights of importance assigned to application's predicted

- Input data items
  (Weight = 4)
- Output report items
  (Weight = 5)
- Inquiry/selection items by user
  (Weight = 4)
- Data file modification/updates
  (Weight = 10)
- Interfaces to other applications
  (Weight = 7)

This produces a raw score that is adjusted by the overall system complexity of the application.

Feature Points

Function-point model was set up for management information applications

What about applications that are rich in complexity with relatively little user interaction, or data-file management, or input/output? What about applications that are, conversely, rich in interaction or input/output?

- Real time systems
- Embedded software
- Process control software
- Communication control

A new item (algorithm) with a new weight (depending on algorithm complexity) is added

- May be based a table for class of algorithm
  - e.g. sorting vs. process control
- May be based on components of algorithm
  - loops, decisions, recursions, etc.
- May be based on Cyclomatic complexity

The data-file weight is downgraded.

System complexity adjustments may be revised.

Language Tables

Feature/function point values are adjusted to a figure of Source Lines of Code (SLOC) (the figure is abstract/fictional)

Each language has its own adjustment
(its so-called "language level")

LANGUAGE LEVEL	PRODUCTIVITY AVERAGE PER STAFF MONTH
1 - 3	5 to 10 Function Points
4 - 8	10 to 20 Function Points
9 - 15	16 to 23 Function Points
16 - 23	15 to 30 Function Points
24 - 55	30 to 50 Function Points
Above 55	40 to 100 Function Points

A few examples

LANGUAGE	LEVEL	AVERAGE SOURCE STATEMENTS PER FUNCTION POINT
Ada 95	6.50	49
ANSI COBOL 85	3.50	91
ANSI SQL	25.00	13
C	2.50	128
C++	6.00	53
MODULA 2	4.00	80

COCOMO

Derives from the early 1980's

Study covered a large mix of applications and companies

Emperical/statistical model

Trys to account for such things as

- product attributes
- computer attributes
- personnel attributes
- project attributes
- process attributes

Product complexity is categorized into three levels

- Simple
- Moderate
- Embedded

Model is a simple exponential

- $\mathrm{A} \times f(\mathrm{SLOC})^{\mathrm{B}} \times \mathrm{M}$

The factors A and B depend on the complexity.

The f value is a sum of a number of factors that weight the SLOC figure by numbers associated with the attributes.