MUNI
FI

COCOMO - COnstructive COst MOdel

PA017 SW Engineering II $\rightarrow$ Aspects of SW Development Management

Jaroslav Ráček **Josef Spurný**

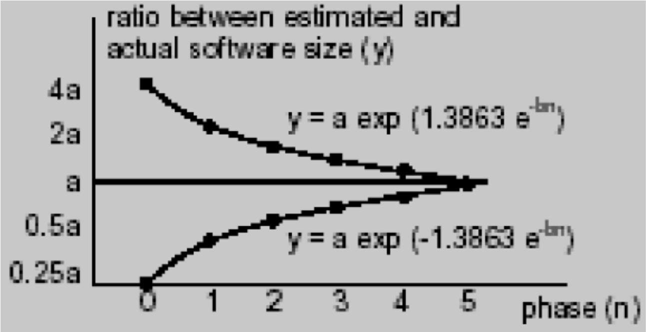
Faculty of Informatics, Masaryk University

October 18, 2022

Core ideas

- Cost for developing an application s directly related to SW size
- Precision of SW size estimation depends on development phase
  - Estimation is more precise in later phases
  - Precision of estimation may differ four times (4:1) both ways
    - 25 KLOC $\leftarrow$ 100 KLOC $\rightarrow$ 400 KLOC
- For some programmers, estimating KLOC is easier than estimating man-days
- For COCOMO, we assume knowledge of KLOC for estimate software cost

Core ideas



ratio between estimated and
actual software size (y)
4a
2a
a
0.5a
0.25a
y = a exp (1.3863 e^-bn)
y = a exp (-1.3863 e^-bn)
0
1
2
3
4
5
phase (n)

COCOMO

- COCOMO - Constructive Cost Model
- B. Boehm (IBM) - 1981, 1984, 1995
- #SLOC as a main SW size and complexity indicator
- Empirical relaionships $E = E(KSCLOC^{1})$ and $T = T(KSLOC)$ where
  - E – effort (work expenditure, man-days, man-months)
  - T – time for development (calendar months)
- Estimation are usually done for larger projects, hence KLOC

[1] Kilo (1000) SourceCode Lines of Code

COCOMO

Sources of empirical data:

- larger number of previous complex projects
  - different kind of projects with different objectives
  - different languages, technologies, development environments
- Data from multiple senior team members (managers, tech. leaders)

This data will be used to set-up parameters for COCOMO.

The better input data, the more precise estimation (output, KLOC)

Original COCOMO

How precise will the estimation be? $\rightarrow$ 3 levels of detail:

- Basic model: unrefined estimation of E(KSLOC) and T(KSLOC) based on KSLOC estimation
- Intermediate model: including other factors on E(KSLOC) and T(KSLOC) estimation (e.g., what tools are used, how experienced is the team, how mature is PM, etc.)
- Advanced model: similar to Intermediate model but involves development phases

More precise estimation only makes sense when good inputs are available

Original COCOMO

How challenging is the project? $\rightarrow$ 3 development models:

- Organic model: simple, easy-to-deliver projects, usually smaller scope
- Semi-detached model: intermediate level of project difficulty, in-between Organic and Embedded level
- Embedded model: large-scope, challenging projects, require high level of experience and creativity from team

Organic model

- small projects (SW < 50 KSLOC)
- very good understanding of requirements
- little restrictions, high level of freedom when designing interface
- team experienced with delivery of similar projects
- low dependency on specific HW or technology
- minimal need for new algorithms and architectures
- little risk of deadline-shortening

Organic model - examples

- scientific or experimental applications
- simple business models and applications
- simple application for storage management
- simple application for manufacturing process management

Semi-detached model

- middle-sized projects (SW < 300 KSLOC)
- reasonably good understanding of requirements
- significant restrictions for interface design
- team has some experience with delivery of similar projects
- medium level of dependency on specific HW or technology
- medium level of need for new algorithms and architectures
- indispensable risk of deadline-shortening (project buffer is usually included)

Semi-detached model - examples

- project of medium size and complexity (e.g. workflows, user account management included...)
- semi-complex application for storage management
- semi-complex manufacturing process management application

Embedded model

- SW with various size (not only large projects)
- only general outline of requirements is known
- high amount of restrictions, strict requirements for interface design
- team has some experience with delivery of similar projects
- high level of dependency on specific HW or technology
- very high requirements for new algorithms and architectures
- high risk of deadline-shortening (project buffer is usually included)

Embedded model - examples

- ambitious and complex systems
- complex signal and control processing systems

Effort and time

Calculating E(KSLOC) and T(KSLOC):

$$E = a.(KSLOC)^b$$
$$T = c.E^d$$

where a, b, c, d are pre-defined model parameters:

- basic-intermediate-advanced model
- organic-semidetached-embedded model
- $\rightarrow$ 9 possible combinations

Parameter values

Empirical values (obtained via statistics from existing projects) for parameters used to calculate E(KSLOC) and T(KSLOC):

- there are values of a, b, c, d defined for each of 9 combinations of models
- examples:
  - basic model + organic model:
    $$a = 3.0, b = 1.12, c = 2.5, d = 0.35$$
  - intermediate model + semi-detached model:
    $$a = 2.8F_c, b = 1.2, c = 2.5, d = 0.32$$
    where F_c is corrective factor (usually slightly above 1.0)

Over time, the formula remained the same but the calibration of parameters was refined.

Parameter values

Empirical values for parameters used to calculate E(KSLOC) and T(KSLOC) Intervals for parameter values:

- $a \in [2.4, 3.6]$ for basic model
- $a \in [2.8F_c, 3.2F_c]$ for intermediate and advanced model
- $b \in [1.05, 1.20]$
- $c = 2.5$
- $d \in [0.32, 0.38]$

Parameter values

Empirical values for parameters used to calculate E(KSLOC) and T(KSLOC)

- In the basic model, all parameters are constant
- In intermediate and advanced model (for all development models), the value of a depends on F_c, all other parameters are constant
- The corrective factor F_c is a product of 15 attributes (*cost drivers*) specific for development process

Corrective factor

4 Areas of attributes having impact on corrective factor F_c

- attributes of SW product
- HW attributes
- attributes of development team
- project attributes

Corrective factor

Attributes having impact on corrective factor F_c may have values on the following scale:

- very low
- low
- normal
- high
- very high
- extremely high

These values have appropriate discrete numeric values.

Corrective factor - attributes of SW product

- RELY - requested reliability (0.75 - 1.40)
- DATA - database size (0.94 - 1.16)
- CPLX - product complexity (0.70 - 1.65)

Corrective factor - HW attributes

- TIME - time restricted calculation (1.00 - 1.66)
- STOR - memory/disc usage (1.00 - 1.56)
- VIRT - reliability (vulnerability) of virtual machines (0.87 - 1.30)
- TURN - turnaround time (0.87 - 1.15)

Corrective factor - development team

- ACAP - analyst capability (1.46 - 0.71)
- PCAP - programming capability (1.42 - 0.70)
- AEXP - experience with similar projects (1.29 - 0.82)
- VEXP - experience with particular virtual machine (1.29 - 0.90)
- LEXP - experience with particular programming language (1.14 - 0.95)

Corrective factor - project attributes

- MODP - usage of modern programming techniques (1.24 - 0.82)
- TOOL - usage of SW tools (1.24 - 0.83)
- SCED - precise planning (1.23 - 1.10)

Cost drivers overview

#		Cost Drivers	VL	L	NOM	HGH	VH	EH
1	PROD	Reliability	0,75	0,88	1	1,15	1,40	X
2	PROD	Database size	X	0,94	1	1,08	1,16	X
3	PROD	Product complexity	0,70	0,85	1	1,15	1,30	1,65
4	PFRM	Execution time constraints	X	X	1	1,11	1,30	1,66
5	PFRM	Main storage constraints	X	X	1	1,06	1,21	1,56
6	PFRM	Virtual machine volatility	X	0,87	1	1,15	1,30	X
7	PFRM	Computer turnaround time	X	0,87	1	1,07	1,15	X
8	PERS	Analyst capability	1,46	1,19	1	0,86	0,71	X
9	PERS	Applications experience	1,29	1,13	1	0,91	0,82	X
10	PERS	Programmer capability	1,42	1,17	1	0,86	0,70	X
11	PERS	Virtual machine experience	1,21	1,10	1	0,90	X	X
12	PERS	Programming language exp.	1,14	1,07	1	0,95	X	X
13	TOOL	Use of modern progr. Techn.	1,24	1,10	1	0,91	0,82	X
14	TOOL	Use of software tools	1,24	1,10	1	0,91	0,83	X
15	TOOL	Req. development schedule	1,23	1,08	1	1,04	1,10	X

Steps for COCOMO application

The following steps are performed when COCOMO is applied:

- Calculation of nominal effort E_n
- Definition corrective factor F_c
- Definition of actual (refined) effort E
- Calculation of development time T and other factors relevant for project

Original COCOMO

Values a, b, c, d are identical for intermediate and advanced level of model

- for intermediate level, the calculation is applied for whole project
- for advanced level, the calculation is applied for individual phases of development lifecycle

COCOMO for modification of existing application

$$ESLOC = ASLOC(0.4DM + 0.3CM + 0.3IM)/100$$, where:

ESLOC = equivalent SLOC
ASLOC = estimated number of modified SLOC
DM = percentage of modifications in design
CM = percentage of modifications in code
IM = integration effort (percentage of original work)

Evolution of COCOMO2

Motivation:

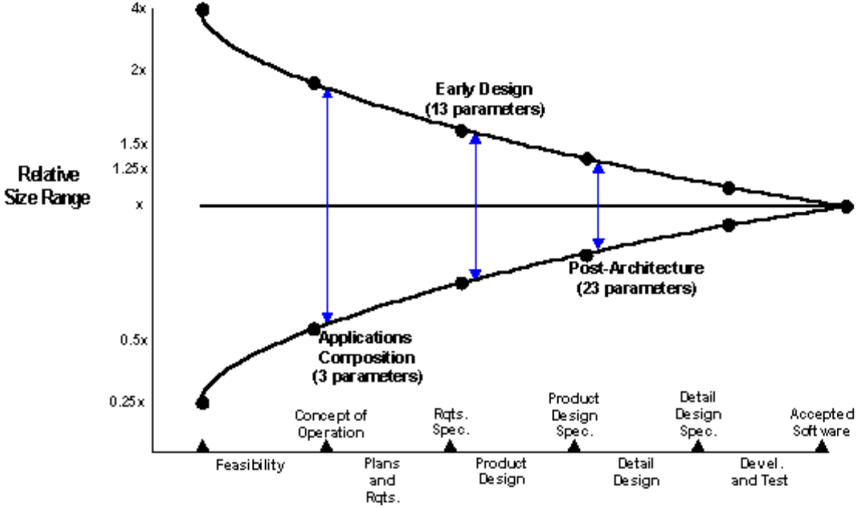
- new SW processes
- new methods for SW size measurement
- new phenomenon of SW re-usability
- need to make decisions on incomplete information

Extended and modified versions

COCOMO2 (1995) - 3 different models, vary by precision / amount of parameters:

- ACM - Application Composition Model
  for projects with modern tools and GUI
- EDM - Early Design Model
  for initial estimations in early project phases
- PAM - Post Architecture Model
  for estimations once architecture is already specified

Extended and modified versions



New factors included in estimation

- RUSE - requested re-usability
- DOCU - documentation created during implementation
- RCPX - complexity and reliability of product
- VMHV - virtual machine variability - host
- VMHP - virtual machine variability - periphery
- PVOL - HW platform variability
- PDIF - HW platform complexity

New factors included in estimation

- PERS - personal capabilities
- PREX - personal experience
- PCON - personal continuity in project
- PEXP - experience with given platform
- LTEX - experience with language and tools
- SECU - security
- SITE - development in multiple sites

Extended and modified versions

COCOMO2 - estimating work expenditure and SW size when modifying existing application:

$$ESLOC = ASLOC(AA + SU + 0.4DM + 0.3CM + 0.3IM)/100, \text{ where:}$$

ESLOC = equivalent SLOC
ASLOC = estimated number of modified SLOC
DM = percentage of modifications in design
CM = percentage of modifications in code
IM = integration effort (percentage of original work)
AA (assessment and assimilation) = work investment needed to determine the extent to which the existing module can be used without modification
SU (software understanding) = readability and understanding of code

MUNI
FACULTY
OF INFORMATICS