Software Effort Estimation Part II

Lecture 6 by Professor Vladimir Geroimenko

Module “Software Project Management“

30 October 2016 - Teaching Week 6

Textbook reference: Chapter 5

Lecture Outline

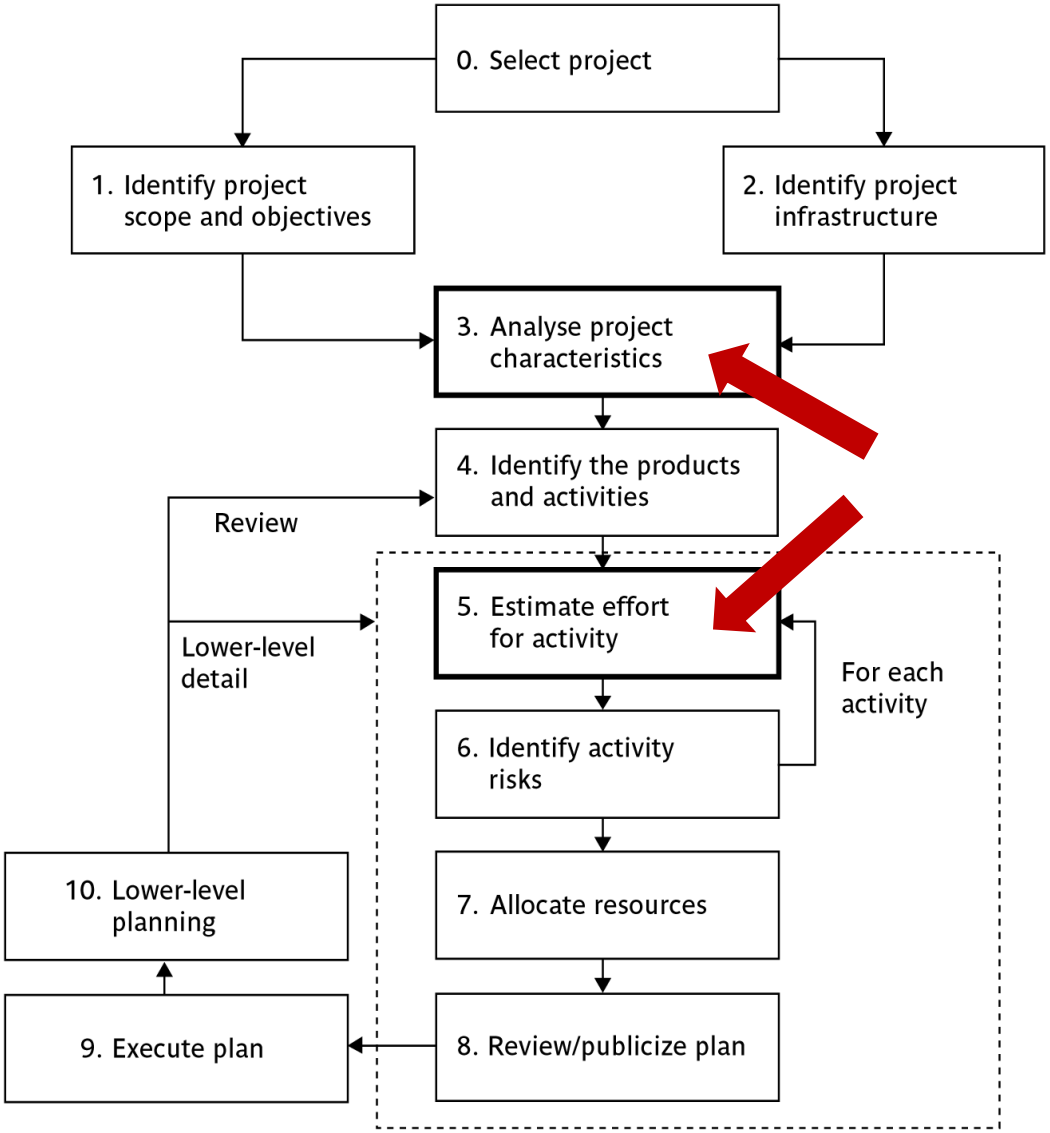
- Introduction to Software Effort Estimation
- Estimation Approaches
  - Top-down estimation
  - Bottom-up estimation
- Estimation Technique
  - Expert judgment
  - Estimation by analogy
  - Algorithmic models
    - Albrecht function points
    - Mark II function points
    - **COCOMO 81 / COCOMO II**

Last week
Part I

This week
Part II

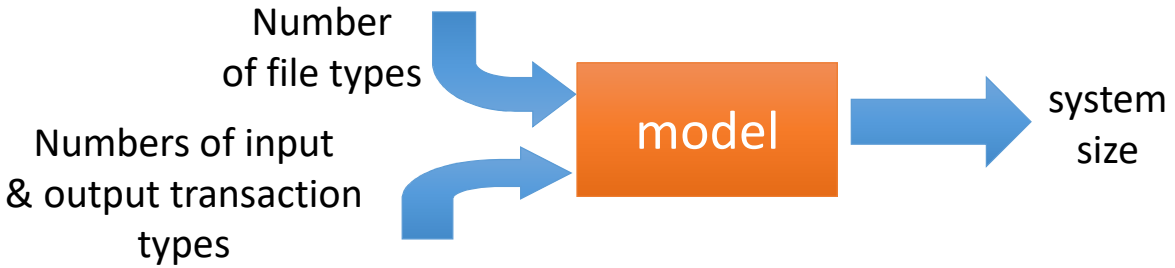
Where we are now in SPMM?

Stepwise Project Management Method (Described in detail in Lecture 2)

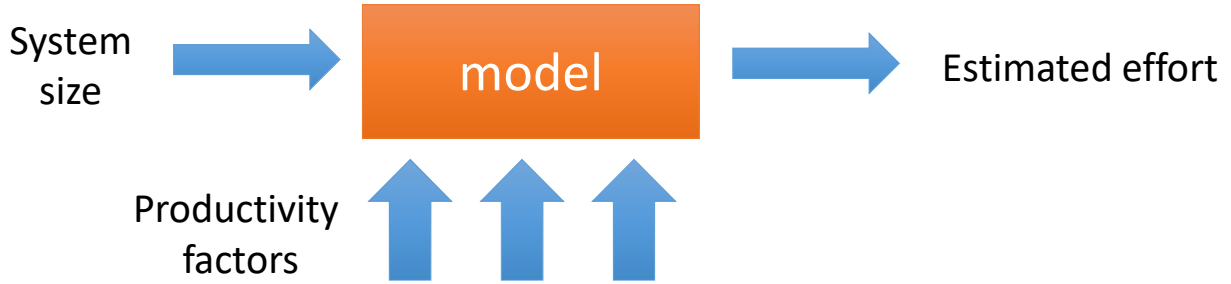


Algorithmic Models

- Based on an algorithm (procedure of calculation)
  - Function Point Methods are used to estimate software ***SIZE***



  - **Co**nstructive **Co**st **Mo**dels (COCOMO) is used to estimate ***EFFORT***



Empirical Model (COCOMO)

COnstructive COst MOdel by Barry W. Boehm,1970

Provide computational means for deriving software cost estimates **as functions of variables (major cost drivers)**.

A group of models:

- Older version – COCOMO81
- Newer version – COCOMO II

COCOMO81

- Based on study of 63 projects
- The basic model was build around the equation

$$effort = c(size)^k$$

effort was measured in ***pm*** or ***person-months*** (1pm = 152 working hours)

size was the system size, measured ***kdsi*** (thousands of delivered source code instructions)

c and **k** were constants, depended on the type of system: organic, semi-detached, or embedded.

COCOMO Modes

- Organic Mode
  - Developed in familiar, stable environment
  - Product similar to previously developed product
  - For example, information systems
- Embedded Mode
  - New product requiring a great deal of innovation
  - Inflexible constraints and interface requirements

    For example, real-time systems
- Semidetached Mode
  - Somewhere between Organic and Embedded

The COCOMO constants

$$effort = c(size)^k$$

System type	c	k
Organic (broadly, information systems)	2.4	1.05
Semi-detached	3.0	1.12
Embedded	3.6	1.20

- *k* exponent – 'to the power of…'
- Adds disproportionately more effort to the larger projects
- Takes account of bigger management overheads

Example

Using COCOMO 81 to estimate effort:

$$effort = c(size)^k$$

1. Calculate function points for the system. For example: FP = 19
2. Calculate source lines of code for the system based on the function points. For development in Java:

   SLOC = 60 x 19 = 1140

3. Apply the basic model for nominal effort estimate. For example: Organic model (c=2.4, k=1.05)

   $effort = c \times size^k = 2.4 \times (1.140)^{1.05} = 2.75$ person-months

COCOMO II

- An updated version of COCOMO 81.
- A family of models that can be used for the estimation at different stages in the SDLC (Software Development Life Cycle).
- It uses different models in 3 different stages of the project: **(1) application composition, (2) early design and (3) post architecture**
- Supports estimation early in the process
- Allows further detailed estimation after the system architecture has been defined
- *We will look specifically at the early design stage.*

COCOMO II Basic Model Equation

Effort = Constant × (Size)$^{scale\ factor}$ × Effort Multipliers

pm = A(size)$^{(sf)}$ ×(em1) ×(em2) ×(em3) …

- Effort in ***pm*** (person-months)
- Constant **A =2.94**
- **size:** Estimated system size in ***kdsi***
- ***sf*** *is the scale factor*
- ***em*** *is an effort multiplier*

sf and em

- **Scale Factor (sf): c**ombined process factors

$$sf = B + 0.01 \times \sum (\text{exponent driver ratings})$$

$$B = 0.91 \text{ (constant)}$$

- **Effort Multiplier (em):** combined effort factors extracted from three predefined tables.

Five Scale Factor Drivers

They are the following five factors which appear to be particularly sensitive to system size:

1. **Precedentedness (PREC).** Degree to which there are past examples that can be consulted.
2. **Development flexibility (FLEX**). Degree of flexibility that exists when implementing the project. (The degree to which the requirement can be met in different ways).
3. **Architecture/risk resolution (RESL).** Degree of uncertainty about requirements.
4. **Team cohesion (TEAM).** The degree to which the team is large and dispersed.
5. **Process maturity (PMAT).** The degree to which the process of software production is structured and organized.

COCOMO II Scale factor values

Driver	Very low	Low	Nominal	High	Very high	Extra high
PREC	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	5.07	4.05	3.04	2.03	1.01	0.00
RESL	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	7.80	6.24	4.68	3.12	1.56	0.00

Example: Scale factor calculation

- A software development team is developing an application which is very similar to previous ones it has developed.
- A very precise software engineering document lays down very strict requirements. **PREC** is very high (score 1.24).
- **FLEX** is very low (score 5.07).
- The good news is that these tight requirements are unlikely to change (**RESL** is high with a score 2.83).
- The team is tightly knit (**TEAM** has high score of 2.19), but processes are informal (so **PMAT** is low and scores 6.24)

Example: Scale factor calculation (cont.)

sf = B + 0.01 x ∑ (exponent driver ratings)

sf = 0.91 + 0.01 × (1.24 + 5.07 + 2.83 + 2.19 + 6.24) = 1.0857

Effort = Constant × (Size)$^{scale\ factor}$ × Effort Multipliers

If system contained **10 kdsi** then estimate would be 2.94 x $10^{1.0857}$ = **35.8 person-months**

Please note: Using exponentiation ('to the power of') adds disproportionately more to the estimates for larger applications

COCOMO II Effort Multipliers

Effort = Constant × (Size)$^{scale\ factor}$ × Effort Multipliers

- Effort multipliers are used to adjust the estimate to take account of productivity factors.

Effort Multipliers for the Early Design Stage

RCPX	Product reliability and complexity
RUSE	Required reusability
PDIF	Platform difficulty
PERS	Personnel capability
PREX	Personnel experience
FCIL	Facilities available
SCED	Schedule pressure

COCOMO II Early Design Effort Multipliers

	Extra low	Very low	Low	Nominal	High	Very high	Extra high
RCPX	0.49	0.60	0.83	1.00	1.33	1.91	2.72
RUSE			0.95	1.00	1.07	1.15	1.24
PDIF			0.87	1.00	1.29	1.81	2.61
PERS	2.12	1.62	1.26	1.00	0.83	0.63	0.50
PREX	1.59	1.33	1.12	1.00	0.87	0.74	0.62
FCIL	1.43	1.30	1.10	1.00	0.87	0.73	0.62
SCED		1.43	1.14	1.00	1.00	1.00	

Example

- A new project is similar in most characteristics to those that an organization has been dealing for some time,

except for

  - the software to be produced is exceptionally complex and will be used in a safety critical system;
  - the software will interface with a new operating system that is currently in the beta status;
  - to deal with this, the team allocated to the job are regarded as exceptionally good, but do not have a lot of experience on this type of software;

Example (cont.)

RCPX	very high	1.91
PDIF	very high	1.81
PERS	extra high	0.50
PREX	nominal	1.00

RCPX	Product reliability and complexity
RUSE	Required reusability
PDIF	Platform difficulty
PERS	Personnel capability
PREX	Personnel experience
FCIL	Facilities available
SCED	Schedule pressure

All other factors are nominal.

Say estimate is **35.8** person-months

With effort multipliers this becomes **35.8** x 1.91 x 1.81 x 0.5 = **61.9** person-months

COCOMO II Post Architecture Effort Multiplier

Modifier type	Code	Effort modifier
Product attributes	RELY	Required software reliability
	DATA	Database size
	DOCU	Documentation match to life-cycle needs
	CPLX	Product complexity
	REUSE	Required reusability
Platform attributes	TIME	Execution time constraint
	STOR	Main storage constraint
	PVOL	Platform volatility
Personnel attributes	ACAP	Analyst capabilities
	AEXP	Application experience
	PCAP	Programmer capabilities
	PEXP	Platform experience
	LEXP	Programming language experience
	PCON	Personnel continuity
Project attributes	TOOL	Use of software tools
	SITE	Multisite development
	SCED	Schedule pressure

TABLE 5.7 COCOMO II Post architecture effort multipliers

Some Attributes

Personnel attributes

- **ACAP** Analyst capability
- **PEXP** Platform experience
- **PCAP** Programmer capability
- **LEXP** Programming language experience
- **AEXP** Application experience

Product attributes

- **RELY** Required software reliability
- **DATA** Database size
- **CPLX** Product complexity

Early Design and Post-Architecture Effort Multipliers

Early Design Cost Driver	Counterpart Combined Post-Architecture Cost Drivers
RCPX	RELY, DATA, CPLX, DOCU
RUSE	RUSE
PDIF	TIME, STOR, PVOL
PERS	ACAP, PCAP, PCON
PREX	AEXP, PEXP, LTEX
FCIL	TOOL, SITE
SCED	SCED

Conclusions

- Estimates are extremely important at any stage of software project management.
- Collect as much information about previous projects as possible.
- Use more than one method of estimating.
- Top-down approaches should be used at the earlier stages of project planning.
- Bottom-up approaches can be effective at the later stages.
- Be careful about using historic productivity data.
- Seek a range of opinions.
- Document your method of doing estimates and record all your assumptions.

Thank you for your attention

Any questions, please?