ESTIMATION MODELS FOR SOFTWARE DEVELOPMENT

Fernando Brito e Abreu (fba@di.fct.unl.pt)

Universidade Nova de Lisboa (http://www.unl.pt)

QUASAR Research Group (http://ctp.di.fct.unl.pt/QUASAR)

ABSTRACT

- *INTRODUCTION*
- *WHY IS IT HARD TO ESTIMATE?*
- *TRADICIONAL ESTIMATION MODELS*
- *ALGORITHMIC ESTIMATION MODELS*
- *CLOSED MODELS*
- *TOOLS*
- *EUROPEAN PROJECTS*
-

INTRODUCTION

Cost factors:

- hardware (amortization and maintenance)
- development tools (amortization and maintenance)
- training
- transports, communications, energy, consumables
- premises (amortization or rents and maintenance)
- salaries, social security, insurance, fringe benefits

Which among these factors are dependent on schedule ?

Which is the dominating factor?

INTRODUCTION

Basic contradictions:

- managers need cost and schedule estimates to negotiate and produce a plan before the project begins
- engineers need to produce a design model before producing meaningful estimates
- clients want a “forfait” contract from a loosely defined requirements specification, ignoring that it will evolve as project progresses

INTRODUCTION

More contradictions:

➢"Horizontal" software is increasingly cheaper, though it may be very sophisticated

➢Business-dependent software systems are often tailored from scratch (thus very costly)

➢Intermediate solutions exist (e.g. SAP et al.)

Cost-benefit tradeoffs must be carefully considered when considering a new system!

WHY IS IT HARD TO ESTIMATE?

- Requirements specs are insufficiently detailed
- Several sorts of requirements exist (e.g. reliability, performance, security, distribution, ...)
- Individual productivity varies much:
  - as high as 1:18 in codification (typically 1:5)
  - subject to uncontrollable breaks (illnesses, turnover)
- No past experience in similar projects
- Technology used has a great influence (design formalisms, languages, tools)

WHY IS IT HARD TO ESTIMATE?

- Many distinct activities take place:
  - analysis
  - design
  - coding
  - configuration management
  - reviews (several types of deliverables)
  - testing (structural and functional)
  - documentation (user and technical ones)
  - metrics collection and analysis
  - defect management and prevention
  - training

TRADICIONAL ESTIMATION MODELS

TRADICIONAL ESTIMATION MODELS

IF YOU TENDER AGAINST OTHERS:

Price to win -

"Cost automatically adjusts to other proposals."

IF YOU ARE THE SINGLE BIDDER:

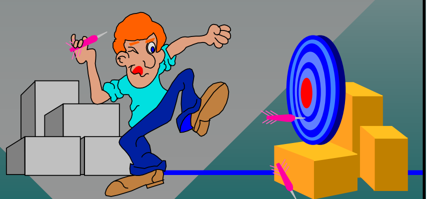
Parkinson Law -

"Work expands to fill the available volume."

TRADICIONAL ESTIMATION MODELS

EXPERTS' OPINION:

- Estimates derived by analogy
- Based on past experience (often gut feeling)
- What can be done when different experts propose distinct estimates?



DELPHI METHOD

Is a consensus achievement technique proposed by Rand Corporation (1948); can be used here

i) Coordinator distributes specification to experts

ii) Each expert, produces separate estimates along with their justification

iii) Coordinator produces a synthesis of estimates (along with the more relevant justifications)

iv) Synthesis is given to experts without discussion and new estimates are asked to them (step ii)

v) Iteration ii), iii) ends when the produced estimates converge enough

ALGORITHMIC ESTIMATION MODELS
W.B.S. based approach

- Is a Bottom-Up estimation method intensively used in engineering (civil, mechanical, etc)

Preconditions:

- development initiative must be decomposed in elementary and independent tasks
- separate estimation for each task

Aggregation of results (used in traditional techniques such as PERT/CPM).

ALGORITHMIC ESTIMATION MODELS
W.B.S. based approach

Effort estimate (each task)

- e_{min} - least possible effort
- e_p – most probable effort
- e_{max} - largest possible effort

Estimated effort

$E_i = (e_{min} + 4e_p + e_{max}) / 6$

Total effort:

$E_T = \sum E_i$

Schedule estimate (each task)

- t_{min} - least possible schedule
- t_p – most probable schedule
- t_{max} - largest possible schedule

Estimated schedule

$T_i = (t_{min} + 4t_p + t_{max}) / 6$

Total schedule:

– *Depends on Critical Path!*

ALGORITHMIC ESTIMATION MODELS
Software specific - early ages (60's/70's)

Heuristic-based:

“in a large project each programmer produces, on the average, 150 lines of tested code source per month".

- *"Each maintenance programmer is able to deal with, on average, four card boxes" (2000 cards)*

ALGORITHMIC ESTIMATION MODELS
Software specific - early models (80's)

- COCOMO Model
- Putnam model
- SLIM model
- Function Points

All models need to be **calibrated** to meet each project specificities:

– development environment, human factor, …

ALGORITHMIC ESTIMATION MODELS
COCOMO (COnstructive COst MOdel)

Proposed by Barry Boehm [Boehm81]

– Initially based on 63 projects done at TRW.

– According to author allows to produce estimates with error less of 20% in 70% of projects.

Interest groups:

– COCOMO Users' Group (USA), since 1985.

– EuroCOCOMO in Europe

ALGORITHMIC ESTIMATION MODELS
COCOMO (COnstructive COst MOdel)

Three development modes:

- Organic
- Half-detached
- Embedded

Three stages:

- Basic Model
- Intermediate Model
- Full Model

ALGORITHMIC ESTIMATION MODELS
COCOMO - Basic Model

Effort Estimate:

$$E = a\,(L)^b$$ [man.month]

	a	b
Organic Mode	2.4	1.05
Half-detached model	3.0	1.12
Embedded Mode	3.6	1.20

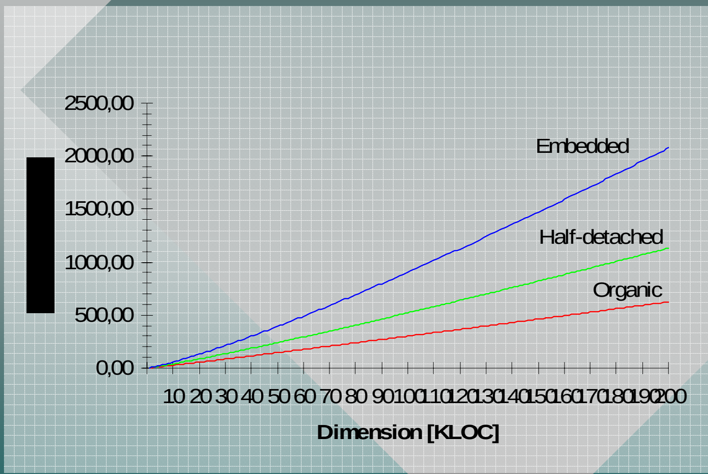
where:

L - lines of code [KLOC]

a, b - calibrating constants

Note: 1 man.month = 152 man.hours

ALGORITHMIC ESTIMATION MODELS
COCOMO - Basic Model



ALGORITHMIC ESTIMATION MODELS
COCOMO - Basic Model

Nominal (optimal) **Schedule**:

$$T = 2.5\ E^{c}\ \text{[months]}$$

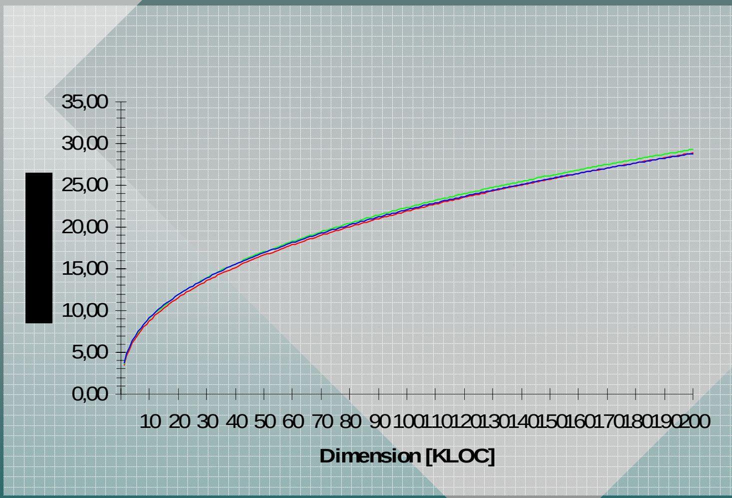
where:

E - effort [men.month]

c - calibrating constant

	c
Organic Model	0.38
Half-detached model	0.35
Embedded Mode	0.32

ALGORITHMIC ESTIMATION MODELS
COCOMO - Basic Model



 5/6/2005

ALGORITHMIC ESTIMATION MODELS
COCOMO - Basic Model

- *Excessively expanded* schedule - Parkinson Law

  According to B. Boehm, expanding schedule to 150% of nominal, increases costs in 110%.

- *Excessively short* schedule can be impracticable. There is a limit for compression!

  According to B. Boehm, compressing schedule to 75% of nominal, increases costs in 125%.

 5/6/2005

ALGORITHMIC ESTIMATION MODELS
COCOMO - Intermediate Model

Factors Influencing Cost (Ci)	Acronym	Very Low	Low	Average	High	Very High	Extra High
• ***relative to product***							
Required reliability level	RELY	0.75	0.88	1.00	1.15	1.40	
Data base dimension	DATA		0.94	1.00	1.08	1.16	
Product complexity	CPLX	0.70	0.85	1.00	1.15	1.30	1.65
• ***relative to computational support***							
Restrictions to execution time	TIME			1.00	1.11	1.30	1.66
Restrictions to storage space	STOR			1.00	1.06	1.21	1.56
Virtual machine volatility	VIRT		0.87	1.00	1.15	1.30	
Response time	TURN		0.87	1.00	1.07	1.15	
• ***relative to human factor***							
Analysts capability	ACAP	1.46	1.19	1.00	0.86	0.71	
Experience in the application domain	AEXP	1.29	1.13	1.00	0.91	0.82	
Programmers capability	PCAP	1.42	1.17	1.00	0.86	0.70	
Experience with virtual machine	VEXP	1.21	1.10	1.00	0.90		
Experience with programming language	LEXP	1.14	1.07	1.00	0.95		
• ***relative to process***							
Adoption of best programming practices	MODP	1.24	1.10	1.00	0.91	0.82	
Use of best tools	TOOL	1.24	1.10	1.00	0.91	0.83	
Accordance to defined schedule	SCED	1.23	1.08	1.00	1.04	1.10	

ALGORITHMIC ESTIMATION MODELS
COCOMO - Intermediate Model

Effort Estimate (new version)**:**

$$E = a(L)^b \prod_{i=1}^{15} c_i$$

where:

c_i - cost drivers

	a
Organic Model	3.2
Haf-detached model	3.0
Embedded Mode	2.8

Note: don't you think there is something strange in the new values of "a"?

ALGORITHMIC ESTIMATION MODELS
COCOMO - Full Model

System decomposition

- Systems are decomposed in subsystems
- Sub-systems may be heterogeneous (each with a specific development mode and specific cost drivers)
- estimates calculated separately for each sub-system

- Division of effort and schedules per phase

ALGORITHMIC ESTIMATION MODELS
COCOMO - Full Model

Division of effort and schedules per phase:

- Influence of each cost driver is split per phase
  - e.g. programmers capability influences mostly the coding phase!

Ratings must be calibrated with past experience

- usual intervals:

Phase	Effort	Schedule
Plan and Analysis	6 - 8%	10 - 40%
Design	16 - 18%	19 - 38%
Code	48 - 68%	24 - 64%
Integration and Test	16 - 34%	18 - 34%

ALGORITHMIC ESTIMATION MODELS
COCOMO II

- Includes 3 stages:
  - **stage 1** - estimation in prototyping or in integration (composition) efforts
  - **stage 2** - estimation in the preliminary design phase (in which the global architecture is defined), when little might be known about the "*cost drivers*"
  - **stage 3** - estimation in post architectural phases

 5/6/2005

ALGORITHMIC ESTIMATION MODELS
COCOMO II

- Systems composed by two types of components:
  - Developed from scratch (KNSLOC)
  - Preexistent ones (KASLOC)
    - AT - % incorporated "as is" (verbatim reuse)
    - 1-AT - % requiring adaptations:
      - assessment and assimilation factor (AA)
      - understandability factor (SU)
      - % design requiring modifications (DM)
      - % code requiring modifications (CM)
      - % of test battery requiring modification (IM)

$$S = KNSLOC + [KASLOC.(1 - AT).(AA + SU + 0,4.DM + 0,3.CM + 0,3.IM)]$$

 5/6/2005

ALGORITHMIC ESTIMATION MODELS
COCOMO II

Effort estimate (man.months):

$$E = A.\left[(1+BRAK).S\right]^{B}.\prod_{i=1}^{17} EM_i + \frac{KASLOC.AT}{ATPROD}$$

where:

- *A, B* - calibrating constants
- *BRAK* - % code thrown away due to requirements volatility
- *EM* - Effort Multipliers (original ones + new ones)
- *ATPROD* - productivity in component integration

A = 2,5 (initial, typical value)

$$B = 1{,}01 + 0{,}01\sum_{j=1}^{5} SF_j$$

ALGORITHMIC ESTIMATION MODELS
COCOMO II

Scale factors:

Acronym	**Description**
PREC	Degree of familiarity with problem and solution domains
FLEX	Flexibility in development
RESL	Risk elimination degree
TEAM	Development team cohesion (ease of interaction)
PMAT	CMM Software Process Maturity

Acronym	Very low	Low	Nominal	High	Very High	Extra High
PREC	4,05	3,24	2,43	1,62	0,81	0,00
FLEX	6,07	4,86	3,64	2,43	1,21	0,00
RESL	4,22	3,38	2,53	1,69	0,84	0,00
TEAM	4,94	3,95	2,97	1,98	0,99	0,00
PMAT	4,54	3,64	2,73	1,82	0,91	0,00

ALGORITHMIC ESTIMATION MODELS
COCOMO II

Schedule estimate (months):

$$T = 3{,}0 . E'^{\,(0{,}33+0{,}2.(B-1{,}01))} . SCED\%$$

where:

SCED% - percentage of compression | expansion on schedule

ALGORITHMIC ESTIMATION MODELS
COCOMO - Tools

- COCOMO II *University of Southern California*
  - http://sunset.usc.edu/j_cocomo/cocomo.html
  - Supports COCOMO II (public domain tool)
- SWAN [Williams91]
  - *IIT Research Institute, Lanham, MD, USA*
  - Also supports Function Points
- COSTAR V7
  - *SOFTSTAR Systems, USA*
  - http://www.softstarsystems.com
  - Supports COCOMO II
- Before You Leap
  - *Remarkable Software, New Zealand*
  - http://www.remarkable.co.nz/byl.htm
  - Also supports Function Points

ALGORITHMIC ESTIMATION MODELS
Function Points

Proposed by Alan Albrecht, then at IBM (mid 70's)

- IFPUG - International Function Point Users' Group since 1986 (Westerville, Ohio, EUA)
  - publishes standards and guides for counting rules
- EFPUG - European Function Point Users' Group (Holland)

Allows to estimate size/complexity of a sw system from the number of visible characteristics (functionalities)

Can be applied earlier (than COCOMO) in the life cycle, from a detailed requirements specification

Is supposed to be independent of the used technology

ALGORITHMIC ESTIMATION MODELS
Function Points

To calculate the system size the following software system characteristics are considered:

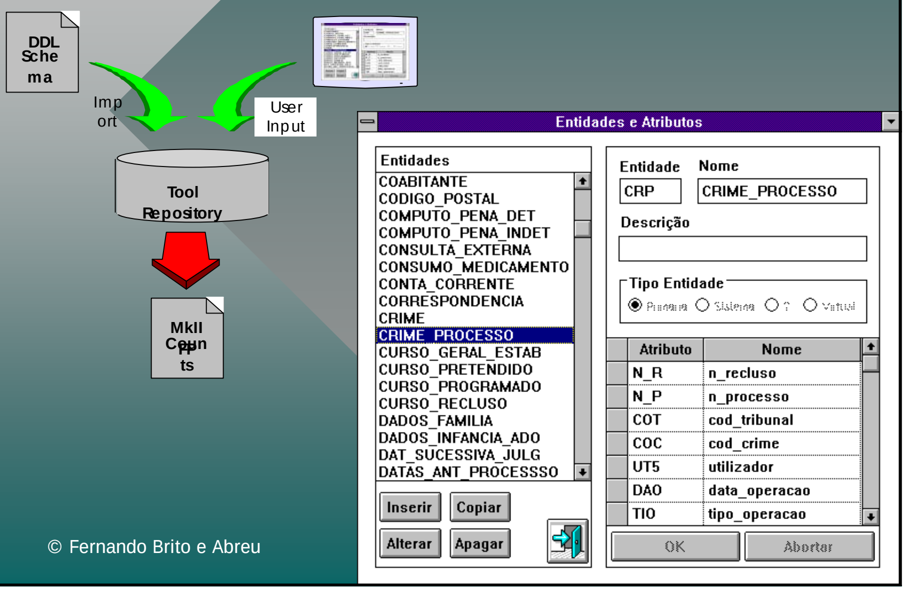
- N. of external inputs (input/modification data forms)
- N. of external outputs (output forms and listings)
- N. of queries (user input/output transactions)
- N. of internal logical files (accessible by the user)
- N. of external interfaces (with other systems, other applications, etc)

ALGORITHMIC ESTIMATION MODELS
Function Points

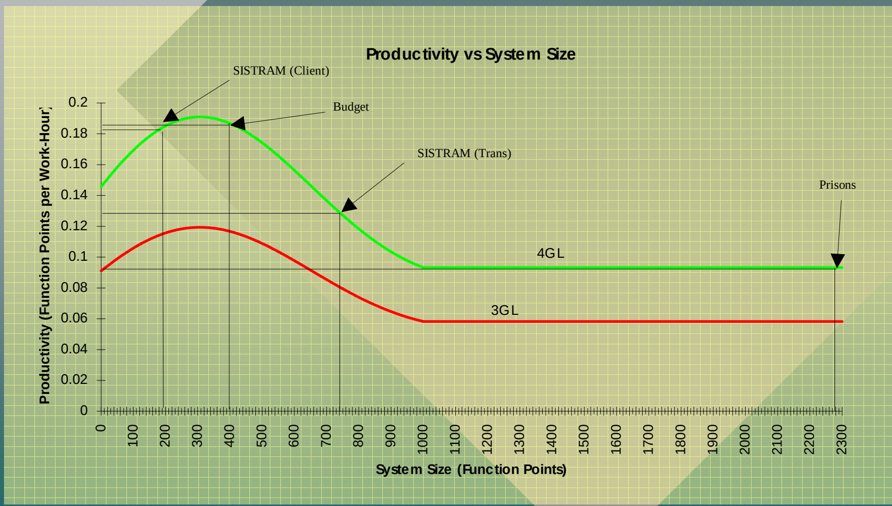
Weighting is done according to complexity criteria

- type and size of input/output files
- number of input fields in a form

ALGORITHMIC ESTIMATION MODELS
Function Points



ALGORITHMIC ESTIMATION MODELS
Function Points



ALGORITHMIC ESTIMATION MODELS
Function Points

Language	LOC per FP
Assembly	320
Macro Assembly	213
C	125 a 128
COBOL (ANSI)	91 a 105
FORTRAN, ALGOL, JOVIAL	100 a 105
PASCAL	85 a 91
MODULA-2	80
PL/1	75
ADA	71
BASIC (ANSI)	64 a 90
LISP	64
APL	32
OBJECTIVE-C	27
SMALLTALK	21