COCOMO - Constructive Cost Modeling

The COCOMO model

- A family of empirical models based on analysis of projects of different companies
- Long history from COCOMO-81 (1981) up to COCOMO-II (1999, 2000)
- Extended to cover different development processes and other aspects, such as quality (COQUALMO)

The COCOMO model

- COCOMO is based on a physical measure (source lines of code)
- Estimations become more precise as we move with development
- Estimation errors:
  - Initial estimations can be wrong by a factor of 4x
  - As we move with the development process, estimations become more precise (and the model takes into account more detailed parameters)

COCOMO: General Structure

$$\text{OUTPUT} = A \cdot (\text{size})^{B} \cdot M$$

- All COCOMO models have the same basic structure
- OUTPUT can be effort or time
- The fundamental measure is code size
(expressed in source lines of code)
- Code size has an exponential effect on effort and size
(although very close to 1)
- Various adjustment factors are used to make the model more precise

COCOMO 81

COCOMO 81: Introduction

- Combination of three models with different levels of detail and complexity:
  - **BASIC**: quick estimation, early development stage
  - **INTERMEDIATE**: more accurate, needs some product characteristics, more mature development stage
  - **ADVANCED**: most detailed, requires more information
- In all COCOMO models:
  - 1 person month = 152 work-hours
  - SLOC is DSI (delivered source instructions) (only the code delivered to the client. E.g. unit testing, conversion code, utilities, ... do not count)

COCOMO 81: Types of Projects

- COCOMO 81 distinguishes among three different types of projects:
  - **ORGANIC**
    - small teams, familiar environment, well-understood applications, simple non-functional requirements (**EASY**)
  - **SEMI DETACHED**
    - project team may have experience mixture, system may have more significant non-functional constraints, organization may have less familiarity with application (**HARDER**)
  - **EMBEDDED**
    - tight constraints, including local regulations and operational procedures; unusual for team to have deep application experience (**HARD**)

COCOMO 81: Basic Model

$$PM = A_{PM} \cdot (KSLOC)^{B_{PM}}$$

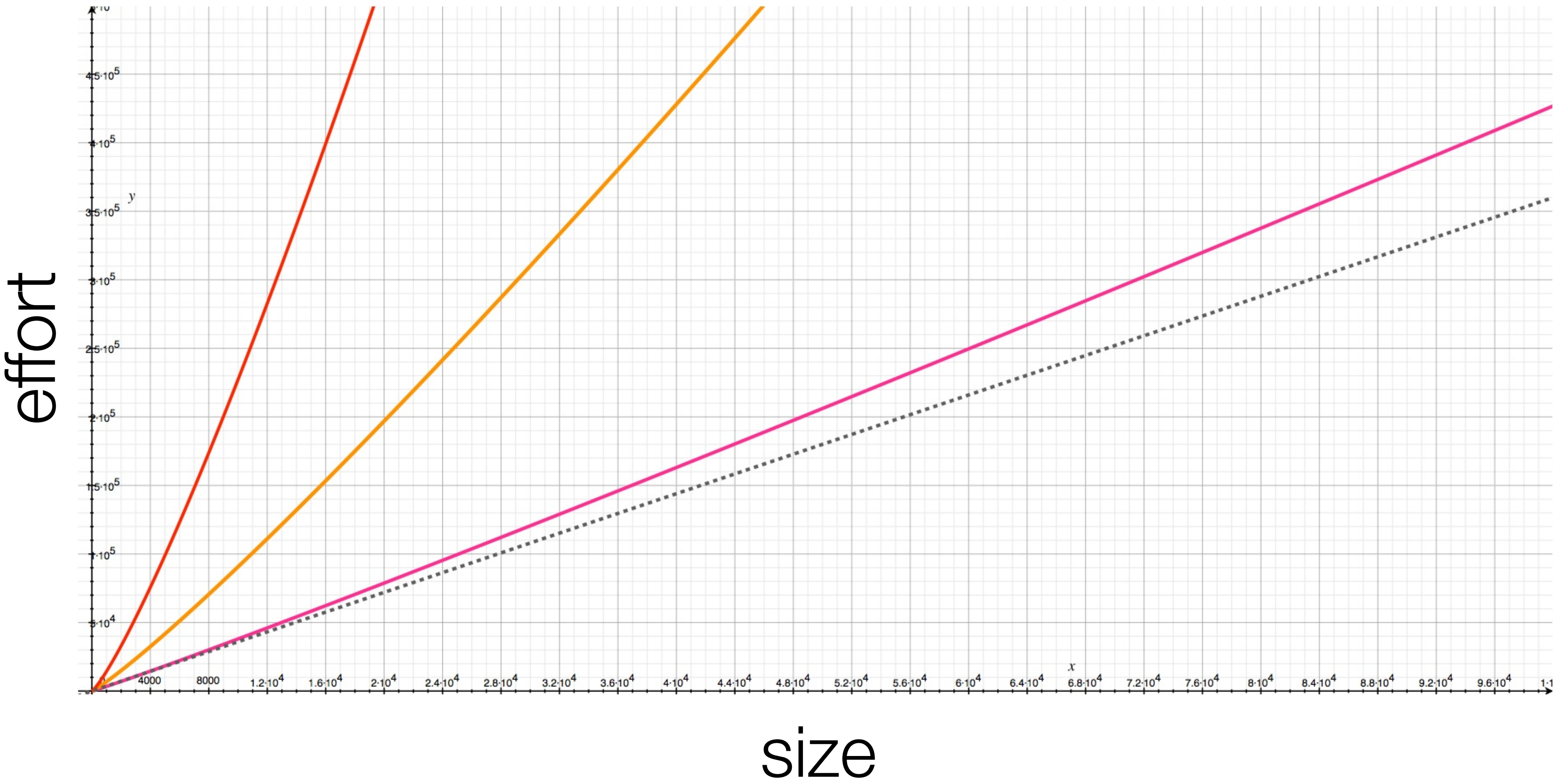
$$TDEV = A_{TDEV}(PM)^{B_{TDEV}}$$

where

- KSLOC: thousands of delivered source lines of code
- M is equal to 1 (and therefore it does not appear in the formulae)

	A	B	A	B
Organic	2.4	1.05	2.5	0.38
Semi-detached	3	1.12	2.5	0.35
Embedded	3.6	1.2	2.5	0.32

COCOMO exponential effect (vs. linear)



Application Example

- Estimation of 50 KDSI for an organic project
  - PM = 2.4 (50)^$^{1.05}$ ~= 146 mm
  - TDEV = 2.5 (371.54)^$^{0.38}$ ~= 16 month
  - Team = 371.54 / 23.69 ~= 9 person
- The effect of different project parameters

	A	B	A	B	PM	TDEV	Team
Organic	2.4	1.05	2.5	0.38	146	16.6	8.8
Semi-detached	3	1.12	2.5	0.35	240	17.0	14.1
Embedded	3.6	1.2	2.5	0.32	394	16.9	23.3

Intermediate COCOMO

- It uses a more fine grained characterization, which uses **attributes** (effort multipliers) to take into account:
  - functional and non-functional requirements
  - project attributes
- The **effort multipliers** are organized in 4 classes and 15 sub-items.
- The importance of each attribute is **qualitatively evaluated** between 1 (very low) and 6 (extra high)
- Each value corresponds to multiplier, in the range [0.7, 1.66] (multiplier < 1 implies reduced cost)
- All the values **are multiplied together** to modulate effort

COCOMO 81: Intermediate Model

$$PM_{nominal} = A_{PM} \cdot (KSLOC)^{B_{PM}}$$

$$PM = PM_{nominal} \cdot \Pi_{i=i}^{15} EM_i$$

$$TDEV = A_{TDEV}(PM)^{B_{TDEV}}$$

	A	B	A	B
Organic	**3.2**	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	**2.8**	1.2	2.5	0.32

Intermediate Model: Parameters

Effort Adjustment Factors		Very_Low	Low	Nominal	High	Very_High	Extr_High
Product Attributes							
Required Software Reliability	RELY	0.75	0.88	**1.00**	1.15	1.40	
Database Size	DATA		0.94	**1.00**	1.08	1.16	
Product Complexity	CPLX	0.70	0.85	**1.00**	1.15	1.30	1.65
Computer Attributes							
Execution Time Constraints	TIME			**1.00**	1.11	1.30	1.66
Main Storage Constraints	STOR			**1.00**	1.06	1.21	1.56
Virtual Machine Volatility	VIRT		0.87	**1.00**	1.15	1.30	
Computer Turnaround Time	TURN		0.87	**1.00**	1.07	1.15	
Personnel Attributes							
Analyst Capability	ACAP	1.46	1.19	**1.00**	0.86	0.71	
Applications Experience	AEXP	1.29	1.13	**1.00**	0.91	0.82	
Programmer Capability	PCAP	1.42	1.17	**1.00**	0.86	0.70	
Virtual Machine Experience	VEXP	1.21	1.10	**1.00**	0.90		
Programming Language Experience	LEXP	1.14	1.07	**1.00**	0.95		
Project Attributes							
Use of Modern Programming Practices	MODP	1.24	1.10	**1.00**	0.91	0.82	
Use of Software Tools	TOOL	1.24	1.10	**1.00**	0.91	0.83	
Required Development Schedule	SCED	1.23	1.08	**1.00**	1.04	1.10	

Attributes

- Attributes:

PRODUCT	= RELY * DATA * CPLX
COMPUTER	= TIME * STOR * VIRT * TURN
PERSONNEL	= ACAP * AEXP * PCAP * VEXP * LEXP
PROJECT	= MODP * TOOL * SCED

- The impact of the parameters is between [0.09, 73.28]
- The PM (or team) estimate the values of parameters to predict actual effort
- Example:
  - If the “required software reliability” is low, the predicted effort is 0.88 of the one computed with the basic formula

COCOMO 81: Detailed Model

- The detailed model:
  - has more detailed multipliers for each development phase
  - organizes the parameters hierarchically, to simplify the computation of systems made of several modules
- Projects are organized in four phases:
    - Requirements Planning and Product Design (PRD)
    - Detailed Design (DD)
    - Code and Unit Test (CUT)
    - Integration Test (IT)
- EM are given and estimated per phase
- Phase data is then aggregated to get the total estimation

COCOMO 81: Advanced Model

- Example of parameter:

Cost Driver	**Rating**	**RPD**	**DD**	**CUT**	**IT**
ACAP	Very Low	1.8	1.35	1.35	1.5
	Low	0.85	0.85	0.85	1.2
	Nominal	1	1	1	1
	High	0.75	0.9	0.9	0.85
	Very High	0.55	0.75	0.75	0.7

COCOMO: Maintenance Phase

- The COCOMO model can also be applied to predict effort during system maintenance
  (system maintenance = small *updates* and *repairs* during the operational life of a system)
- Most of development parameters apply both to development and maintenance
  (some do not: SCED, RELY, MODP)
- One essential input is an estimation of the ACT
  (annual change traffic)

COCOMO 81: Maintenance

$$ACT = \frac{\%Added + \%Modified}{100}$$

$$PM = ACT \cdot PM_{nom} \cdot EAF_{maint}$$

COCOMO II

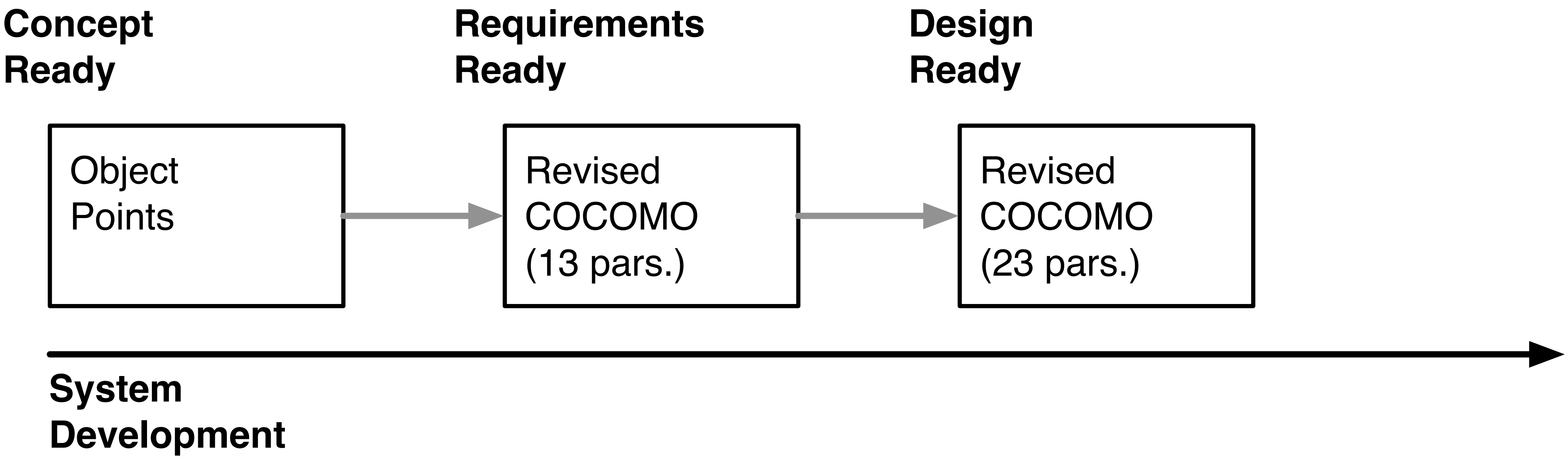
COCOMO II

- COCOMO II builds upon COCOMO 81 to take into account:
  - New development processes (e.g., spiral)
  - Increased flexibility in software development (e.g. reuse, automatic code generation)
  - Need for decision making with incomplete information
  - New data about projects (not really a need, rather an opportunity) (161 projects vs. 61)

COCOMO II: Models

- COCOMO II incorporates a range of sub-models that produce increasingly detailed software estimates.
- The sub-models in COCOMO II are:
  - **Application Composition Model.** For prototyping
  - **Early Design Model.** Used when requirements are available but design has not yet started.
  - **Post-architecture model.** Used once the system architecture has been designed and more information about the system is available.
- Moreover:
  - **Reuse model.** Used to compute the effort of integrating reusable components.

COCOMO II: Model Stages



COCOMO II: ED and PA Models

$$PM_{NS} = 2.94 \cdot (SIZE)^{E} \cdot \Pi_{i=1}^{n} EM_i$$

$$E = 0.91 + 0.01 * \sum_{j=1}^{5} \mathrm{SF}_i$$

$$TDEV_{NS} = 3.67 * (PM_{NS})^{F}$$

$$F = 0.28 + 0.01 * \sum_{j=1}^{5} \mathrm{SF}_i$$

All constants can (need to) be adjusted with organization-dependent values.

(Strongly recommended: 2.94, effort multiplier and 3.67, schedule multiplier)

The difference between ED and PA is the number of parameters

The exponent depends on adjustment factors
(rather than being just a constant as in COCOMO '81)

COCOMO II: Effort Multipliers

- From 7 (Early Design) to 17 (Post Architecture) according to the level of detail needed
- For instance:

	Early Design cost drivers	Post-Architecture cost drivers (Counterpart combined)
Product reliability and complexity	RCPX	RELY, DATA, CPLX, DOCU
Required reuse	RUSE	RUSE
Platform difficulty	PDIF	TIME, STOR, PVOL
Personnel capability	PERS	ACAP, PCAP, PCON
Personnel experience	PREX	AEXP, PEXP, LTEX
Facilities	FCIL	TOOL, SITE
Required Development Schedule	SCED	SCED

Source: http://www.ifi.uzh.ch/req/courses/seminar_ws02/reports/Seminar_4.pdf

COCOMO II: Scale Factors

- The exponent is computed by providing qualitative answers to the following factors:
  - **Precedentedness**: how novel the project is for the organization
  - **Flexibility**: development flexibility (e.g. rigidity of compliance to requirements)
  - **Design/Risk**: thoroughness of design and risk resolution
  - **Team Cohesion**
  - **Process Maturity:** maturity with respect to the CMMI questionnaire

Scale Factors (W_i)	Very Low	Low	Nominal	High	Very High	Extra High
PREC	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar	largely familiar	throughly familiar
FLEX	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
RESL	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
TEAM	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions
PMAT	Weighted average of "Yes" answers to CMM Maturity Questionnaire					

Source: http://www.ifi.uzh.ch/req/courses/seminar_ws02/reports/Seminar_4.pdf

Algorithmic
Techniques
Conclusions

COCOMO Considerations

- A series of progressively more complex models
- COCOMO computes both D and E and manpower is derived from D and E.
  (we often estimate E, decide M, and compute only D)
- Project cost estimates may be self-fulfilling: the estimate defines the budget and the product is adjusted to meet the budget
- A precise application requires organizations to setup their own measurement programs (to fine-tune parameters)
- Models need to be adapted to changing technologies and the technology changes fast... it might be difficult to keep it up to date

Algorithmic Techniques: Recap

- Based on system characteristics and productivity metrics collected about past projects
- Different models
  - Function Points:
    Req ☞ UFP ☞ FP, MM/FP

    Req ☞ UFP ☞ SLOC/UFP and COCOMO
  - Object Points
    Screens, Reports, Modules ☞ NOP, NOP/Month
  - COCOMO
    SLOC ☞ PDEV, TDEV ☞ Team Size = PDEV / TDEV

The Improvement “factory”