Lecture 15 – The COCOMO Cost Model (Sommerville Ch. 23)

Karl R. Wilcox
K.R.Wilcox@reading.ac.uk

Objectives

- To describe the COCOMO 2 algorithmic cost estimation model

The COCOMO model

- An empirical model based on project experience
- Well-documented, 'independent' model which is not tied to a specific software vendor
- Long history from initial version published in 1981 (COCOMO-81) through various instantiations to COCOMO 2
- COCOMO 2 takes into account different approaches to software development, reuse, etc.

COCOMO I

- Mainframe focused
- Aimed at government projects
- Assumed 4 attributes of projects
  - Product
  - Computer
  - Personnel
  - Project
- And 15 cost drivers based on these attributes
- Basic measure was “Source Lines of Code”

Cost Drivers - 1

- Product
  - Required software reliability
  - Database size
  - Software product complexity
- Computer
  - Execution time
  - Main storage
  - Virtual machine volatility
  - Computer turnaround time

Cost Drivers - 2

- Personnel
  - Analyst & programmer capability
  - Applications, language & VM experience
- Project
  - Modern programming practices
  - Software tools
  - Timescale constraints

Also considered

- Project size (S, M, I, L, VL)
- Mode of organisation
- Expected rates of change
- Security requirements
- Workday schedule

The COCOMO Process

- Data collection
  - To get details of each cost driver
- Entry into “spreadsheet” like model (or custom package)
  - Using a published, widely validated algorithm
- End result is estimate of lines of code required
- Could then turn this into expected schedule

COCOMO Problems

- Mainframe based
- Waterfall model assumed
  - Poor fit for spiral model, rapid development, Objects etc.

- So COCOMO II
  - Replaced about 30% of COCOMO I
  - New & updated cost drivers
  - COTS, 4GL & other models included

COCOMO 2 levels

- COCOMO 2 is a 3 level model that allows increasingly detailed estimates to be prepared as development progresses
- Early prototyping level
  - Estimates based on object points and a simple formula is used for effort estimation
- Early design level
  - Estimates based on function points that are then translated to LOC
- Post-architecture level
  - Estimates based on lines of source code

Early prototyping level

- Supports prototyping projects and projects where there is extensive reuse
- Based on standard estimates of developer productivity in object points/month
- Takes CASE tool use into account
- Formula is
  - PM = (NOP × (1 - %reuse/100)) / PROD
  - PM is the effort in person-months, NOP is the number of object points and PROD is the productivity

Object point productivity

Developer's experience and capability	Very low	Low	Nominal	High	Very high
ICASE maturity and capability	Very low	Low	Nominal	High	Very high
PROD (NOP/month)	4	7	13	25	50

Early design level

- Estimates can be made after the requirements have been agreed
- Based on standard formula for algorithmic models
  - $PM = A \times Size^{B} \times M + PM_m$ where
  - $M = PERS \times RCPX \times RUSE \times PDIF \times PREX \times FCIL \times SCED$
  - $PM_m = (ASLOC \times (AT/100)) / ATPROD$
  - A = 2.5 in initial calibration, Size in KLOC, B varies from 1.1 to 1.24 depending on novelty of the project, development flexibility, risk management approaches and the process maturity

Multipliers

- Multipliers reflect the capability of the developers, the non-functional requirements, the familiarity with the development platform, etc.
  - RCPX - product reliability and complexity
  - RUSE - the reuse required
  - PDIF - platform difficulty
  - PREX - personnel experience
  - PERS - personnel capability
  - SCED - required schedule
  - FCIL - the team support facilities
- PM reflects the amount of automatically generated code

Post-architecture level

- Uses same formula as early design estimates
- Estimate of size is adjusted to take into account
  - Requirements volatility. Rework required to support change
  - Extent of possible reuse. Reuse is non-linear and has associated costs so this is not a simple reduction in LOC
  - ESLOC = ASLOC × (AA + SU +0.4DM + 0.3CM +0.3IM)/100
    - ESLOC is equivalent number of lines of new code. ASLOC is the number of lines of reusable code which must be modified, DM is the percentage of design modified, CM is the percentage of the code that is modified , IM is the percentage of the original integration effort required for integrating the reused software.
    - SU is a factor based on the cost of software understanding, AA is a factor which reflects the initial assessment costs of deciding if software may be reused.

The exponent term

- This depends on 5 scale factors (see next slide). Their sum/100 is added to 1.01
- Example
  - Precedenteness - new project - 4
  - Development flexibility - no client involvement - Very high - 1
  - Architecture/risk resolution - No risk analysis - V. Low - 5
  - Team cohesion - new team - nominal - 3
  - Process maturity - some control - nominal - 3
- Scale factor is therefore 1.17

Exponent scale factors

Scale factor	Explanation
Precedentedness	Reflects the previous experience of the organisation with this type of project. Very low means no previous experience, Extra high means that the organisation is completely familiar with this application domain.
Development flexibility	Reflects the degree of flexibility in the development process. Very low means a prescribed process is used; Extra high means that the client only sets general goals.
Architecture/risk resolution	Reflects the extent of risk analysis carried out. Very low means little analysis, Extra high means a complete a thorough risk analysis.
Team cohesion	Reflects how well the development team know each other and work together. Very low means very difficult interactions, Extra high means an integrated and effective team with no communication problems.
Process maturity	Reflects the process maturity of the organisation. The computation of this value depends on the CMM Maturity Questionnaire but an estimate can be achieved by subtracting the CMM process maturity level from 5.

Multipliers

- **Product attributes**
  - concerned with required characteristics of the software product being developed
- **Computer attributes**
  - constraints imposed on the software by the hardware platform
- **Personnel attributes**
  - multipliers that take the experience and capabilities of the people working on the project into account.
- **Project attributes**
  - concerned with the particular characteristics of the software development project

Project cost drivers

Product attributes			
RELY	Required system reliability	DATA	Size of database used
CPLX	Complexity of system modules	RUSE	Required percentage of reusable components
DOCU	Extent of documentation required		
Computer attributes			
TIME	Execution time constraints	STOR	Memory constraints
PVOL	Volatility of development platform		
Personnel attributes			
ACAP	Capability of project analysts	PCAP	Programmer capability
PCON	Personnel continuity	AEXP	Analyst experience in project domain
PEXP	Programmer experience in project domain	LTEX	Language and tool experience
Project attributes			
TOOL	Use of software tools	SITE	Extent of multi-site working and quality of site communications
SCED	Development schedule compression		

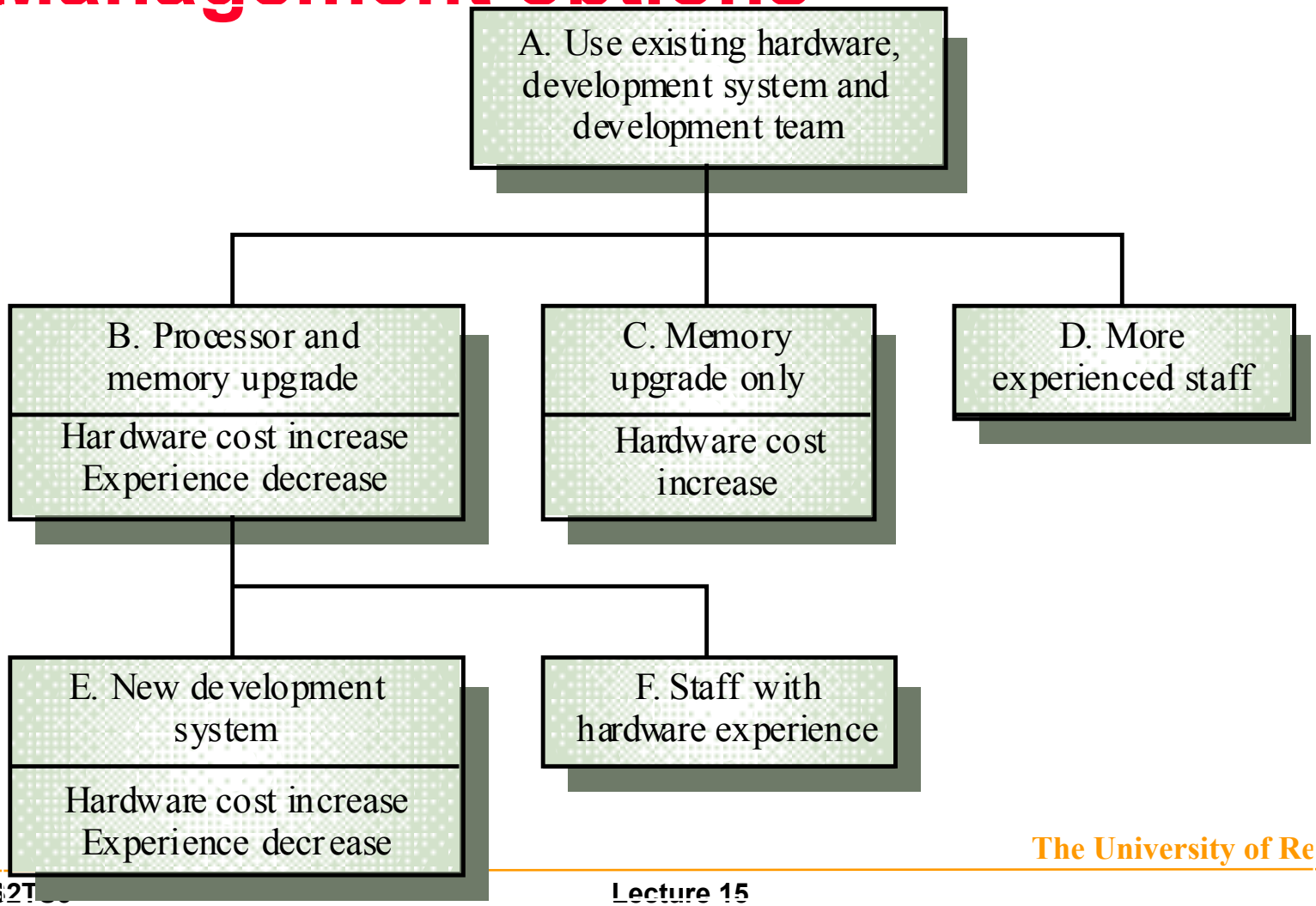
Effects of cost drivers

Exponent value System size (including factors for reuse and requirements volatility) **Initial COCOMO estimate without cost drivers**	1.17 128, 000 DSI **730 person-months**
Reliability Complexity Memory constraint Tool use Schedule **Adjusted COCOMO estimate**	Very high, multiplier = 1.39 Very high, multiplier = 1.3 High, multiplier = 1.21 Low, multiplier = 1.12 Accelerated, multiplier = 1.29 **2306 person-months**
Reliability Complexity Memory constraint Tool use Schedule **Adjusted COCOMO estimate**	Very low, multiplier = 0.75 Very low, multiplier = 0.75 None, multiplier = 1 Very high, multiplier = 0.72 Normal, multiplier = 1 **295 person-months**

Project planning

- Algorithmic cost models provide a basis for project planning as they allow alternative strategies to be compared
- Embedded spacecraft system
  - Must be reliable
  - Must minimise weight (number of chips)
  - Multipliers on reliability and computer constraints > 1
- Cost components
  - Target hardware
  - Development platform
  - Effort required

Management options



Management options costs

Option	RELY	STOR	TIME	TOOLS	LTEX	Total effort	Software cost	Hardware cost	Total cost
A	1.39	1.06	1.11	0.86	1	63	949393	100000	1049393
B	1.39	1	1	1.12	1.22	88	1313550	120000	1402025
C	1.39	1	1.11	0.86	1	60	895653	105000	1000653
D	*1.39*	*1.06*	*1.11*	*0.86*	*0.84*	*51*	*769008*	*100000*	*897490*
E	1.39	1	1	0.72	1.22	56	844425	220000	1044159
F	1.39	1	1	1.12	0.84	57	851180	120000	1002706

Option choice

- Option D (use more experienced staff) appears to be the best alternative
  - However, it has a high associated risk as experienced staff may be difficult to find
- Option C (upgrade memory) has a lower cost saving but very low risk
- Overall, the model reveals the importance of staff experience in software development

Project duration and staffing

- As well as effort estimation, managers must estimate the calendar time required to complete a project and when staff will be required
- Calendar time can be estimated using a COCOMO 2 formula
  - $TDEV = 3 \times (PM)^{(0.33+0.2*(B-1.01))}$
  - PM is the effort computation and B is the exponent computed as discussed above (B is 1 for the early prototyping model). This computation predicts the nominal schedule for the project
- The time required is independent of the number of people working on the project

Staffing requirements

- Staff required can't be computed by diving the development time by the required schedule
- The number of people working on a project varies depending on the phase of the project
- The more people who work on the project, the more total effort is usually required
- A very rapid build-up of people often correlates with schedule slippage

Key points

- The COCOMO model takes project, product, personnel and hardware attributes into account when predicting effort required
- Algorithmic cost models support quantitative option analysis
- The time to complete a project is not proportional to the number of people working on the project