CISC 322

Software Architecture

Lecture 20:

Software Cost Estimation 2

Emad Shihab

Slides adapted from Ian Sommerville and Ahmed E. Hassan

Estimation Techniques

- There is no simple way to make accurate estimates of the effort required
  - Initially, not much detail is given
  - Technologies and people may be unknown

- Project cost estimates may be self-fulfilling
  - Estimate defines budget, project adjusted to meet budget

Many Estimation Techniques

- Algorithmic cost modeling
- Expert judgment
- Estimation by analogy
- Parkinson's Law
- Pricing to win

Algorithmic code modelling

- Model is built based on historical cost information
- Generally based on the size of the software

Expert judgement

- Several experts in software development and the application domain are consulted
- Process iterates until some consensus is reached
- Advantages: Relatively cheap estimation method. Can be accurate if experts have direct experience of similar systems
- Disadvantages: Very inaccurate if there are no experts!

Estimation by analogy

- The project is compared to a similar project in the same application domain
- Advantages: Accurate if project data available
- Disadvantages: Impossible if no comparable project has been tackled

Parkinson's Law

- “Work expands to fill the time available” i.e., the project costs whatever resources are available
- Advantages: No overspending
- Disadvantages: System is usually unfinished

Pricing to win

- The project costs whatever the customer has to spend on it
- Advantages: You get the contract
- Disadvantages: The probability that the customer gets the system he or she wants is small. Often, costs do not accurately reflect the work required

Cost Estimation Approaches

- The aforementioned techniques may be used top-down or bottom-up
- **Top-down**: Starts at the system level and assess system functionality and its delivery through subsystems
- **Bottom-up**: Start at component level and aggregate to obtain system effort

Top-down vs. Bottom-up

- Top-down:
  - Usable without much knowledge
  - Factors in integration, configuration and documentation costs
  - Can underestimate low-level problems
- Bottom-up:
  - Usable when architecture of the system is known
  - May underestimate system-level activities such as integration

Algorithmic Cost Modeling

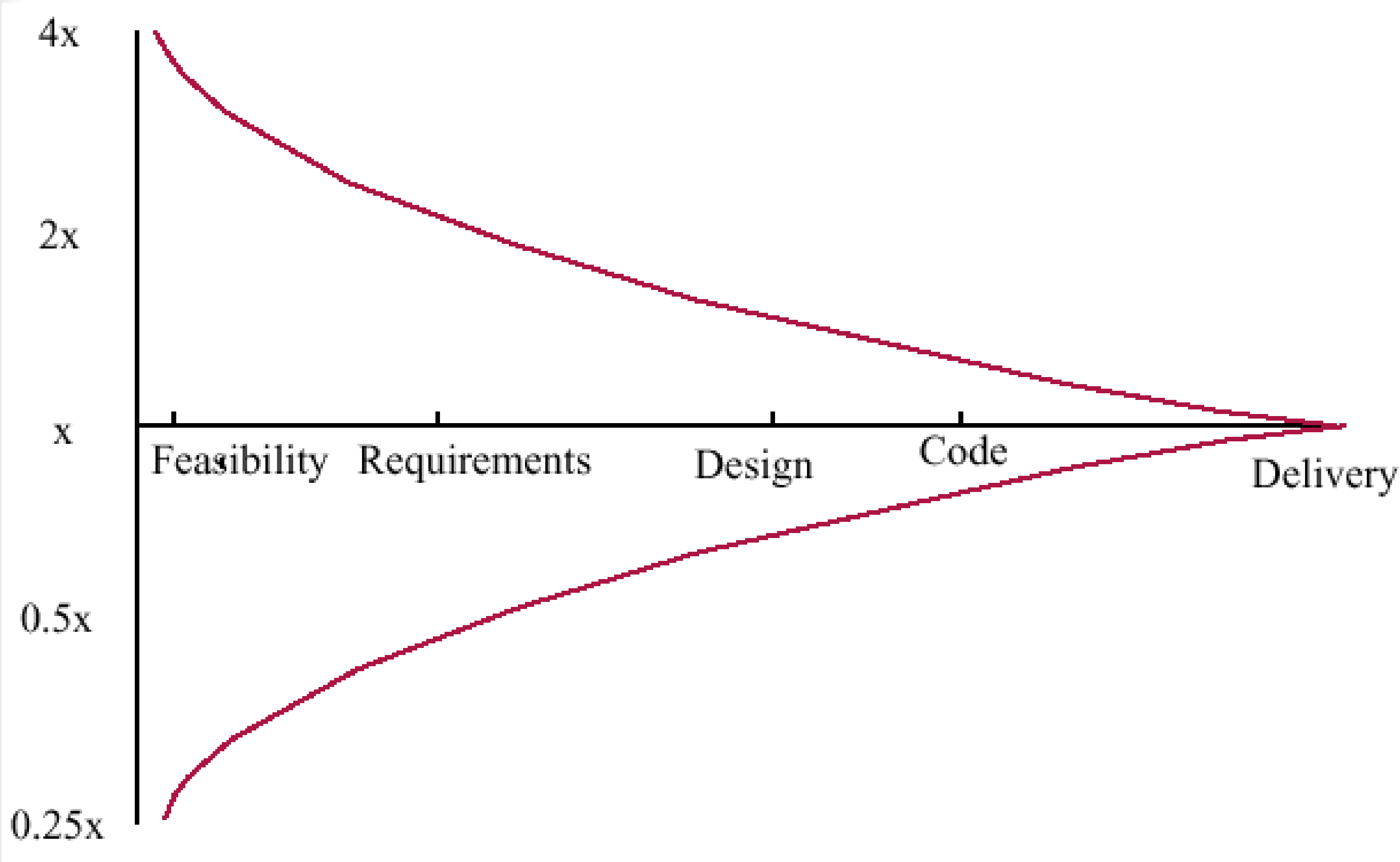
- A cost model can be built by analyzing the cost and attributes of similar projects
- **Effort** = A x Size^{B} x M
- **A** – depends on organization
- **B** – ~1-1.5 reflects disproportionate effort for large projects (comm. and conf. management)
- **M** – reflects product, process and people attributes

Estimation Accuracy

- Difficult to estimate size early on. B and M are subjective
- Several factors influence the final size
  - Use of COTS and components
  - Programming language
- Estimations become more accurate as development progresses

[Sommerville 2000]

Estimate uncertainty



COCOMO Model

- Empirical model based on project experience
- Started with COCOMO-81 and later revised to COCOMO 2
- COCOMO 2 is very detailed and takes into account different approaches, reuse, etc…

COCOMO 81

Project complexity	Formula	Description
Simple	$PM = 2.4\ (KDSI)^{1.05} \times M$	Well-understood applications developed by small teams.
Moderate	$PM = 3.0\ (KDSI)^{1.12} \times M$	More complex projects where team members may have limited experience of related systems.
Embedded	$PM = 3.6\ (KDSI)^{1.20} \times M$	Complex projects where the software is part of a strongly coupled complex of hardware, software, regulations and operational procedures.

A – depends on organization
B – reflects disproportionate effort for large projects
M - reflects product, process and people attributes

COCOMO 2 levels

- **Early prototyping model**
  - Estimates based on OP and a simple formula
- **Early design model**
  - Estimates based on FP that are translated to LOC
- **Reuse model**
  - Estimates effort to integrate reused and generated code
- **Post-architecture level**
  - Estimates based on lines of source code

Early Prototyping Level

- Supports prototyping projects and projects where software is developed by composing existing components

- PM = (NOP x (1 - %reuse/100)) / PROD
  - PM is the effort in person-months
  - NOP is the number of object points
  - 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 requirements
- Based on standard algorithmic model
  - $PM = A \times Size^B \times M$
    - A = 2.94 in initial calibration
    - Size in KLOC (aprox. from FP)
    - B varies from 1.1 to 1.24 depending on novelty, development flexibility, risk management and the process maturity
    - M = PERS x RCPX x RUSE x PDIF x PREX x FCIL x SCED

Multipliers

- Multipliers developers, non-functional requirements, 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

The Reuse Model

- Effort is required to integrate automatically generated code
- PM_{Auto} = (ASLOC x (AT/100)) / ATPROD
  - ASLOC – No. LOC that have to be adapted
  - AT - % of adapted code that is automatically generated
  - ATPROD – engineer productivity in adapting code (2400 LOC/month)
- e.x., 20,000 LOC, 30% automatically generated
  - (20,000 x 30/100) / 2400 = 2.5 pm

Post-architecture level

- Uses same formula as early design estimates ($PM = A \times Size^B \times M$)
- Size estimate for the software should be more accurate at this stage. Takes into consideration:
  - New code to be developed
  - Rework required to support change
  - Extent of possible reuse

The exponent term (B)

- This depends on 5 scale factors. Their sum/100 is added to 1.01

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.

The Exponent Term (B) Example

- 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

Multipliers (M)

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

- Product attributes
  - 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

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 Duration

- COCOMO
  - $TDEV = 3 \times (PM)^{(0.33+0.2*(B-1.01))}$
- COCOMO 2
  - $TDEV = 3 \times (PM)^{(0.33+0.2*(B-1.01))} \times SCEDP/100$

  - TDEV – calendar days
  - PM – effort
  - B – Exponent
  - SCEDP - % increase or decrease in nominal schedule

COCOMO Example

Function Point Table

Number of FPs	Complexity		
External user type	Low	Average	High
Inputs	3	4	6
Outputs	4	5	7
Files	7	10	15
Interfaces	5	7	10
Queries	3	4	6

Object Point Analysis – Complexity Weighting

	Complexity		
Type of object	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL component	N/A	N/A	10

Object Point Analysis – Productivity Rate

	Very low	Low	Nominal	High	Very High
Developer's experience and capability	4	7	13	25	50
CASE maturity and capability	4	7	13	25	50

COCOMO II

$$\text{Effort} = A \times (\text{Size})^{B} \times M$$

- Effort in terms of person-months
- A: 2.45 in 1998
- Size: Estimated Size in KLOC
- B: combined process factors
- M: combined effort factors

System to be built

- An airline sales system is to be built in C:
  - Back-end database server has already been built.
- We will use object point estimation technique for high level estimates and FP for detailed estimates

Object Point Analysis

- Application will have 3 screens and will produce 1 report:
  - A booking screen: records a new sale booking
  - A pricing screen: shows the rate for each day and each flight
  - An availability screen: shows available flights
  - A sales report: shows total sale figures for the month and year, and compares figures with previous months and years

Rating of system

- Booking screen:
  - Needs 3 data tables (customer info, customer history table, available seats)
  - Only 1 view of the screen is enough. So, the booking screen is classified as simple.
- Similarly, the levels of difficulty of the pricing screen, the availability screen and the sales report are classified as simple, simple and medium, respectively. There is no 3GL component.

Rating Results

Name	Objects	Complexity	Weight
Booking	Screen	Simple	1
Pricing	Screen	Simple	1
Availability	Screen	Medium	2
Sales	Report	Medium	5
		Total	9

- Assessment of the developers and the environment shows:
  - The developers' experience is very low (4)
  - The CASE tool is low (7). So, we have a productivity rate of 5.5.
- The project requires approx. 1.64 (= 9/5.5) person-months.

Function Point Estimation (FP->KLOC)

Name	External user types	Complexity	FP
Booking	External output type	Low	4
Pricing	External inquiry type	Low	3
Availability	External inquiry type	Medium	4
Sales	External output type	Medium	5
		Total	16

FP->LOC

- Total function points = 16
- Published figures for C show that:
  - 1 FP = 128 LOC in C
- Estimated Size
  - 16 * 128 = 2048 = 2 KLOC

Scale Factor Estimation (B)

Name	Very low (0.05)	Low (0.04)	Nominal (0.03)	High (0.02)	Very High (0.01)	Extra High (0.00)	**Assessment**	**Value**
Precedentedness	Thoroughly unprecedented	Largely unprecedented	Somewhat unprecedented	Generally familiar	Largely familiar	Thoroughly familiar	**Very high**	**0.01**
Flexibility	Rigorous	Occasional relaxation	Some relaxation	General conformity	Some conformity	General goals	**Very high**	**0.01**
Significant risks eliminated	Little (20%)	Some (40%)	Often (60%)	Generally (75%)	Mostly (90%)	Full (100%)	**Nominal**	**0.03**
Team interaction process	Very difficult	Some difficult	Basically cooperative	Largely cooperative	Highly cooperative	Seamless interactions	**High**	**0.02**
Process maturity	Level 1	Level 2	Level 2+	Level 3	Level 4	Level 5	**Low**	**0.04**
							Add	**1.01**
							Total	**1.13**

Effort Adjustment Factors (M)

Identifier	Name	Ranges (VL – EH)	Assessment VL/L/N/H/VH/EH	Values
RCPX	product Reliability and ComPleXity	0.5 – 1.5	low	0.75
RUSE	required reusability	0.5 – 1.5	nominal	1.0
PDIF	Platform DIFficulty	0.5 – 1.5	high	1.1
PERS	PERSonnel capability	1.5 – 0.5	high	0.75
PREX	PeRsonnel EXperience	1.5 – 0.5	very high	0.65
FCIL	FaCILities available	1.5 – 0.5	nomial	1.0
SCED	SChEDule pressure	1.5 – 0.5	low	1.2
			Product	0.4826

- Effort = $2.94 \times (2.048)^{1.13} \times 0.4826 = 3.19$ person-months

References

- Hughes, B., and Cotterell, M. (1999) *Software project management*, 2nd ed., McGraw Hill
- Pfleeger, S.L. (1998) *Software Engineering: Theory and Practice*, Prentice Hall
- Royce, W. (1998) *Software Project Management: A Unified Framework*, Addison Wesley
- Center for Software Engineering, USC (1999) *COCOMO II Model Definition Manual*.