Software Project Management

Week – 9

Today's Lecture

- Project Cost Analysis

✓Organizational structure of a software house

✓Configuration Management

Slides derived from Ian Somerville, PMP Prep and B Boehm

2

Fundamental estimation questions

- How much effort is required to complete an activity?
- How much calendar time is needed to complete an activity?
- What is the total cost of an activity?
- Project estimation and scheduling and interleaved management activities

Software cost components

- Hardware and software costs
- Travel and training costs
- Effort costs (the dominant factor in most projects)
  - salaries of engineers involved in the project
  - Social and insurance costs
- Effort costs must take overheads into account
  - costs of building, heating, lighting
  - costs of networking and communications
  - costs of shared facilities (e.g library, staff restaurant, etc.)

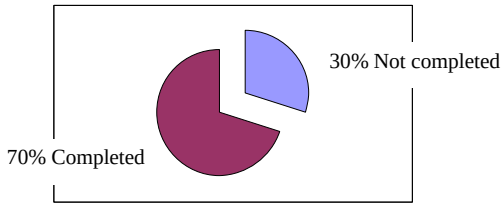
Software cost components

- Hardware and software costs
- Travel and training costs
- **Effort costs (the dominant factor in most projects)**
  - **salaries of engineers involved in the project**
  - **Social and insurance costs**
- Effort costs must take overheads into account
  - costs of building, heating, lighting
  - costs of networking and communications
  - costs of shared facilities (e.g library, staff restaurant, etc.)

What is Software Cost Estimation

- Predicting the cost of resources required for a software development process

Software is a Risky Business



- 53% of projects cost almost 200% of original estimate.
- Estimated $81 billion spent on failed U.S. projects in 1995.
  - All surveyed projects used waterfall lifecycle.

Software is a Risky Business

- British Computer Society (BCS) survey:
  - 1027 projects
  - Only 130 were successful !
- Success was defined as:
  - deliver all system **requirements**
  - within **budget**
  - within **time**
  - to the **quality** agreed on

Costing and Pricing

- Estimates are made to discover the cost of development for producing a software system
- There is not a simple relationship between the development cost and the price charged to the customer
- Broader organisational, economic, political and business considerations influence the price charged

Why early Cost Estimation?

- Cost estimation is needed early for s/w pricing
- S/W price = cost + profit

Programmer productivity

- A measure of the rate at which individual engineers involved in software development produce software and associated documentation
- Not quality-oriented although quality assurance
  is a factor in productivity assessment
- Essentially, we want to measure useful functionality produced per time unit

Productivity measures

- Size related measures based on some output from the software process. This may be lines of delivered source code, object code instructions, etc.
- Function-related measures based on an estimate of the functionality of the delivered software. Function-points are the best known of this type of measure

Measurement problems

- Estimating the size of the measure
- Estimating the total number of programmer months which have elapsed
- Estimating contractor productivity (e.g. documentation team) and incorporating this estimate in overall estimate

Fundamental estimation questions

- **Effort**
  - How much effort is required to complete an activity?
  - Units: man-day (person-day), man-week, man-month,,..
- **Duration**
  - How much calendar time is needed to complete an activity? Resources assigned
  - Units: hour, day, week, month, year,..
- **Cost of an activity**
  - What is the total cost of an activity?
- Project estimation and scheduling are interleaved management activities

Software Cost Components

1. Effort costs (dominant factor in most projects)
   - salaries
   - Social and insurance & benefits
2. Tools costs: Hardware and software for development
   - Depreciation on relatively small # of years 300K US$
3. Travel and Training costs (for particular client)
4. Overheads(OH): Costs must take overheads into account
   - costs of building, air-conditioning, heating, lighting
   - costs of networking and communications (tel, fax,)
   - costs of shared facilities (e.g library, staff restaurant, etc.)
   - depreciation costs of assets
   - Activity Based Costing (ABC)

S/W Pricing Policy

S/W price is influenced by

- economic consideration
- political consideration
- and business consideration

Software Pricing Policy/Factors

Factor	Description
Market opportunity	A development organisation may quote a low price because it wishes to move into a new segment of the software market. Accepting a low profit on one project may give the opportunity of more profit later. The experience gained may allow new products to be developed.
Cost estimate uncertainty	If an organisation is unsure of its cost estimate, it may increase its price by some contingency over and above its normal profit.
Contractual terms	A customer may be willing to allow the developer to retain ownership of the source code and reuse it in other projects. The price charged may then be less than if the software source code is handed over to the customer.
Requirements volatility	If the requirements are likely to change, an organisation may lower its price to win a contract. After the contract is awarded, high prices may be charged for changes to the requirements.
Financial health	Developers in financial difficulty may lower their price to gain a contract. It is better to make a small profit or break even than to go out of business.

Measurement problems

- Estimating the size of the measure
- Estimating the total number of programmer-months which have elapsed
- Estimating contractor productivity (e.g. documentation team) and incorporating this estimate in overall estimate

Lines Of Code (LOC)

- Program length (LOC) can be used to predict program characteristics e.g. person-month effort and ease of maintenance
- What's a line of code?
  - The measure was first proposed when programs were typed on cards with one line per card
  - How does this correspond to statements as in Java which can span several lines or where there can be several statements on one line?
- What programs should be counted as part of the system?
- Assumes linear relationship between system size and volume of documentation

Versions of LOC

- DSI : Delivered Source Instructions
- KLOC Thousands of LOC
- DSI
  - One instruction is one LOC
  - Declarations are counted
  - Comments are not counted

LOC

- Advantages
  - Simple to measure
- Disadvantages
  - Defined on code: it can not measure the size of specification
  - Based on one specific view of size: length.. What about complexity and functionality !!
  - Bad s/w may yield more LOC
  - Language dependent
- Therefore: Other s/w size attributes must be included

LOC Productivity

- The lower level the language, the less productive the programmer
  - The same functionality takes more code to implement in a lower-level language than in a high-level language
- Measures of productivity *based on LOC* suggest that programmers who write verbose code are more productive than programmers who write compact code !!!

FUNCTION POINT

Function points

- Based on a combination of program characteristics
  - external inputs and outputs
  - user interactions
  - external interfaces
  - files used by the system
- A weight is associated with each of these
- The function point count is computed by multiplying each raw count by the weight and summing all values

Function points

- Function point count modified by complexity of the project
- FPs can be used to estimate LOC depending on the average number of LOC per FP for a given language
  - LOC = AVC * number of function points
  - AVC is a language-dependent factor varying from 200-300 for Assembly language to 2-40 for a 4GL
- FPs are very subjective. They depend on the estimator.
  - Automatic function-point counting is impossible

Software Size Estimation Approach

- Function Points Approach
  - Popularized by the extensive research of A.J. Albrecht and J.E. Gaffney
    - Related article: "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation"
  - Used in industry
  - Considered better than LOC
  - Measures Software Size from specification
    - Early in the life Cycle, unlike LOC.

26

Function point Method

The function point estimation process:

- **Stage 1**: Compute the unadjusted (crude) function count (points) **(UFC / CFP).**
- **Stage 2**: Compute the relative complexity adjustment factor **(RCAF)** for the project. RCAF varies between 0 and 70.
- **Stage 3**: Compute the number of function points **(FP):**

  FP = CFP x (0.65 + 0.01 x RCAF)

Function Point Method

- Relates to the following five software components:
  - *Number of user inputs*
  - *Number of user outputs*
  - *Number of user online queries*
  - *Number of logical files*
  - *Number of external interface*
- Weighted factors are applied to each components according to their complexity.

Function Point Calculation Form

Software system components	Complexity level									Total CFP
	Simple			average			complex			
	Count	Weight Factor	Points	Count	Weight Factor	Points	Count	Weight Factor	Points	
	A	B	C= AxB	D	E	F= DxE	G	H	I= GxH	
User inputs		3			4			6		
User outputs		4			5			7		
User online queries		3			4			6		
Logical files		7			10			15		
External interfaces		5			7			10		
Total CFP										

Relative Complexity Adjustment Factor

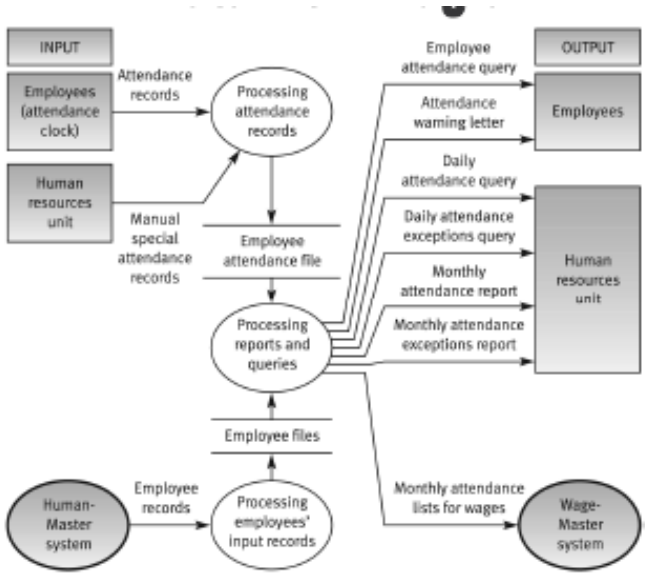
- Summarizes the complexity characteristics of the software system
  - assign grades (0 to 5) to the 14 subjects that substantially affect the development effort:
  - RCAF = $\sum_{i=1}^{14} S_i$
- The RCAF determines the *technical complexity factor* (TCF): *TCF = 0.65 + 0.01 * RCAF*
- FP = CFP * TCF

Relative Complexity Adjustment Factor - Form

No	Subject	Grade
1	Requirement for reliable backup and recovery	0 1 2 3 4 5
2	Requirement for data communication	0 1 2 3 4 5
3	Extent of distributed processing	0 1 2 3 4 5
4	Performance requirements	0 1 2 3 4 5
5	Expected operational environment	0 1 2 3 4 5
6	Extent of online data entries	0 1 2 3 4 5
7	Extent of multi-screen or multi-operation online data input	0 1 2 3 4 5
8	Extent of online updating of master files	0 1 2 3 4 5
9	Extent of complex inputs, outputs, online queries and files	0 1 2 3 4 5
10	Extent of complex data processing	0 1 2 3 4 5
11	Extent that currently developed code can be designed for reuse	0 1 2 3 4 5
12	Extent of conversion and installation included in the design	0 1 2 3 4 5
13	Extent of multiple installations in an organization and variety of customer organizations	0 1 2 3 4 5
14	Extent of change and focus on ease of use	0 1 2 3 4 5
	Total = RCAF	

EXAMPLE

Example – Employee Attendance System



Complexity Adjustment

Software system components	Complexity level									Total CFP
	Simple			average			complex			
	Count	Weight Factor	Points	Count	Weight Factor	Points	Count	Weight Factor	Points	
	A	B	C= AxB	D	E	F= DxE	G	H	I= GxH	
User inputs	1	3	3	---	4	---	1	6	6	9
User outputs	---	4	---	2	5	10	1	7	7	17
User online queries	1	3	3	1	4	4	1	6	6	13
Logical files	1	7	7	---	10	---	1	15	15	22
External interfaces	---	5	---	---	7	---	2	10	20	20
Total CFP										81

RCAF Calculation

No	Subject	Grade
1	Requirement for reliable backup and recovery	0 1 2 3 4 5
2	Requirement for data communication	0 1 2 3 4 5
3	Extent of distributed processing	0 1 2 3 4 5
4	Performance requirements	0 1 2 3 4 5
5	Expected operational environment	0 1 2 3 4 5
6	Extent of online data entries	0 1 2 3 4 5
7	Extent of multi-screen or multi-operation online data input	0 1 2 3 4 5
8	Extent of online updating of master files	0 1 2 3 4 5
9	Extent of complex inputs, outputs, online queries and files	0 1 2 3 4 5
10	Extent of complex data processing	0 1 2 3 4 5
11	Extent that currently developed code can be designed for reuse	0 1 2 3 4 5
12	Extent of conversion and installation included in the design	0 1 2 3 4 5
13	Extent of multiple installations in an organization and variety of customer organizations	0 1 2 3 4 5
14	Extent of change and focus on ease of use	0 1 2 3 4 5
	Total = RCAF	41

FP Calculation

$$FP = CFP \times (0.65 + 0.01 \times RCAF)$$

$$FP = 81 \times (0.65 + 0.01 \times 41) = 85.86$$

- Total cost is calculated by

Software Size Estimation Approach

- Subjectivity in assigning weights
  - Weights were determined subjectively from IBM experience
  - Not necessarily applicable to other environment
- Double counting
  - Internal complexity is counted in giving weights in UFC, and again in TCF
- Researchers have shown that TCF does not improve accuracy over UFC

37

Software Size Estimation Approach

- It does not measure mathematical optimization, various embedded systems, scientific and real-time process control applications as well as it measures information systems
- Problems with measurement theory
  - Incorrectly combines measurement from different scales
  - Weights and TCF ratings are on an ordinal scale
  - Counts are on absolute scale (or at least, a ratio scale)
  - The linear combination of the two are meaningless

38

Software Size Estimation Approach

- Object Points Approach
  - With the advent of Object Oriented (OO) programming/development, there has been a growing demand for new ways to measure software size in an OO environment.
  - According to the Cost Xpert User's manual, object metric based estimating is seeing an increased following among companies using object oriented techniques throughout the software life cycle.

39

Object points

- Object points are an alternative function-related measure to function points when 4Gls or similar languages are used for development
- Object points are NOT the same as object classes
- The number of object points in a program is a weighted estimate of
  - The number of separate screens (forms) that are displayed
  - The number of reports that are produced by the system
  - The number of 3GL modules that must be developed to supplement the 4GL code

OBJECT POINTS

Object Points

- Develop by Banker et al. [Banker et al]
- Originally intended for CASE tool development projects
  - May apply to other projects
- Adapted for Object-Oriented software
- Use object counts instead of function counts

42

Object Points

- An "object" can be:
  - Logical system component (**Rule Sets**)
  - Language-specific constructs (**3GL-Module**)
  - UI component (**Screen Definitions**)
  - User report (**Report**)
- An "object" is NOT just a raw object class
- Conducted at a more macro level than Function Points

43

Object Points - How

Object Type	Object Complexity		
	Simple	Medium	Difficult
Rule Sets			
3GL-Module			
Screen Definition			
User reports			

- Steps
  1. For each object type, count all instances
  2. Each object is assessed a complexity weight

[Fenton et al]

44

Object Points - How

- Steps
  3. Sum up complexity weights of all objects to get the Object-Point (OP)
  4. Multiply OP by a reuse factor (RF)
  5. If this is not a new program, calculate NOP (New Object-Point)

NOP = OP (1 – RF)

[Bohem et al]

45

Object Points - How

- Steps
  6. Based on NOP, determine expected productivity cost

Developers Experience, ICASE maturity / capability	Very Low	Low	Nominal	High	Very High
Productivity cost	4	7	13	25	50

Effort = NOP / Productivity cost

[Bohem et al]

46

Object Points - Variants

- Other variations exist
  - PRICE Object-Points [Ferens]
  - SEER-SEM Object-Points [Ferens]
  - WebObjects [Reifer]
- No established standards
- Some variants have rules for OP-to-FP conversion

47

Object Points - Analysis

- Advantages
  - Same as in Function Points
  - May be easier to visualize and understood by clients
- Disadvantages
  - Less use than Function Points
  - Lack of standards

48

Size Estimation Methods

- Entry Criteria
  - Complete requirement definition for the project/iteration
  - Historical data from similar software, if required by the method
- Exit Criteria
  - Depends on the estimation methods used

49

Size Estimation Methods

- Wideband-Delphi
- PROBE

50

Size Estimation Methods – Wideband-Delphi

- Steps
  1. A group of experts is each given the program's specification and an estimation form
  2. The group of experts and a moderator meet to discuss the product and any estimation issues
  3. Each estimator anonymously complete the estimation forms

51

Size Estimation Methods – Wideband-Delphi

– Steps
  4. Moderator collects completed estimation forms
     – tabulates the results
     – returns them to the experts
     – Only each estimator's personal estimate is identified; all others are anonymous
  5. Experts meet to discuss the results, revising their estimates as they feel appropriate
  6. Repeat Step #1-5 until the estimates converge to an acceptable range.

52

Size Estimation Methods – Wideband-Delphi

- Can be applied at the project or component level
- Usually only examine a small section or component of the overall effort/project [Boehm]

53

Size Estimation Methods – Wideband-Delphi

- Advantages
  - Iterative, team based, collaborative estimating
  - Less biased than individual estimation
  - Does not require historical data
  - Can be used at both high-level and detailed level estimation

54

Size Estimation Methods – Wideband-Delphi

- Disadvantages
  - May be hard to find more than one expert
  - Difficult to repeat with different group of experts
  - Possible to reach consensus on an incorrect estimate, people may not be skeptical enough
  - Can develop a false sense of confidence

55

Size Estimation Methods – Wideband-Delphi

- Disadvantages
  - May fail to reach a consensus
  - Experts may be biased in the same subjective direction
  - Lots of overhead (time, team involvement, planning) for a relatively small sets of tasks
  - Takes quite a few “steps” or iterations
  - Does not require historical data

56

Size Estimation Methods - PROBE

- **PRO**xy **B**ased **E**stimation
- Proxy are used instead of SLOC
  - Function Points
  - Object-Points
  - Other levels of granularity (slides in a presentation, pages in a book, etc.)
- Requires historical data

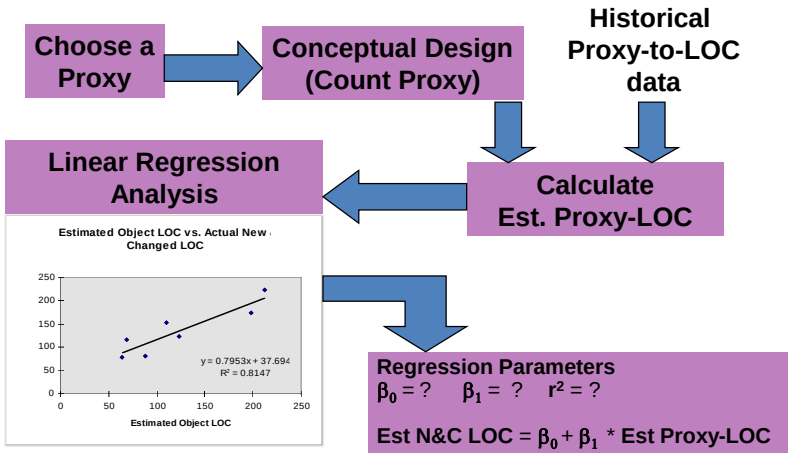
57

Size Estimation Methods - PROBE

- A good proxy should be:
  - Related closely to the effort required to develop the product
  - Automatically counted
  - Visualized easily and early in the project
  - Customizable according to organization needs

58

Size Estimation Methods - PROBE



59

Size Estimation Methods - PROBE

- Advantages
  - Forces you to keep and use historical data
- Disadvantages
  - Doesn't work without historical data
  - Not always have appropriate historical data
  - Not always have enough appropriate historical data
  - Needs lots of historical data for linear regression analysis to work

60

Reliable Size Estimations

- Attributes
  1. Structured
  2. Defined
  3. Applicable for all projects
  4. Applicable throughout a project
  5. Easily adjustable for future projects
  6. Susceptible to statistical analysis
  7. Use of a suitable estimation proxy
  8. Ability to automate

61

Algorithmic code modelling

- A formula – empirical relation:
  - based on historical cost information and which is generally based on the size of the software
- The formulae used in a formal model arise from the analysis of historical data.

Expert Judgement

- One or more experts in both software development and the application domain use their experience to predict software costs. 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!

ALGORITHMIC METHODS

Estimation by Analogy

- Experience-based Estimates
- The cost of a project is computed by comparing
  the project to a similar project in the same application domain
- Advantages: Accurate if project data available
- Disadvantages: Impossible if no comparable project has been tackled. Needs systematically maintained cost database

Pricing to Win

- The project costs whatever the customer budget is.
- Advantages: You get the contract
- Disadvantages:
  - The probability that the customer gets the system he/she wants is
    small.
  - Costs do not accurately reflect the work required

Pricing to Win

- This approach may seem unethical and unbusiness like
- However, when detailed information is lacking it may be the only appropriate strategy
- The project cost is agreed on the basis of an outline proposal and the development is constrained by that cost
- A detailed specification may be negotiated or an evolutionary approach used for system development

Top-down Estimation

- Usable *without* knowledge of the system architecture and the components that might be part of the system
- Takes into account costs such as integration, configuration management and documentation
- Can underestimate the cost of solving difficult low-level technical problems

Bottom-up estimation

- Usable when
  - the architecture of the system is known and
  - components identified
- Accurate method if the system has been designed in detail
- May underestimate costs of system level activities such as integration and documentation

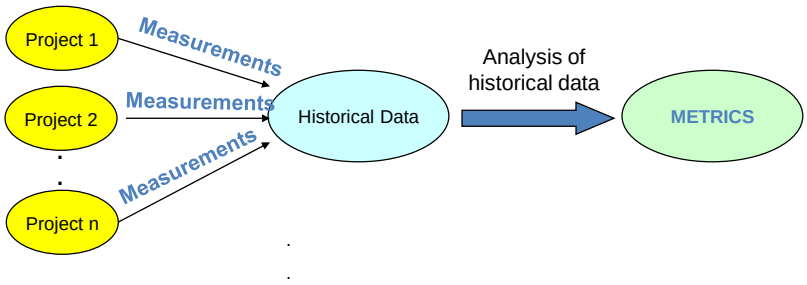
Estimation Methods

- S/W project estimation should be based on several methods
- If these do not return approximately the same result, there is insufficient information available
- Some action should be taken to find out more in order to make more accurate estimates
- Pricing to win is sometimes the only applicable method

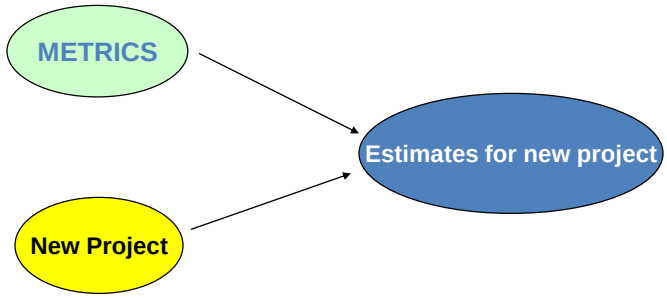
Algorithmic Cost Modelling

- Most of the work in the cost estimation field has focused on algorithmic cost modelling.
- Costs are analysed using mathematical formulas linking costs or inputs with METRICS to produce an estimated output.
- The formula is based on the analysis of historical data.
- The accuracy of the model can be improved by calibrating the model to your specific development environment, (which basically involves adjusting the weighting parameters of the metrics).

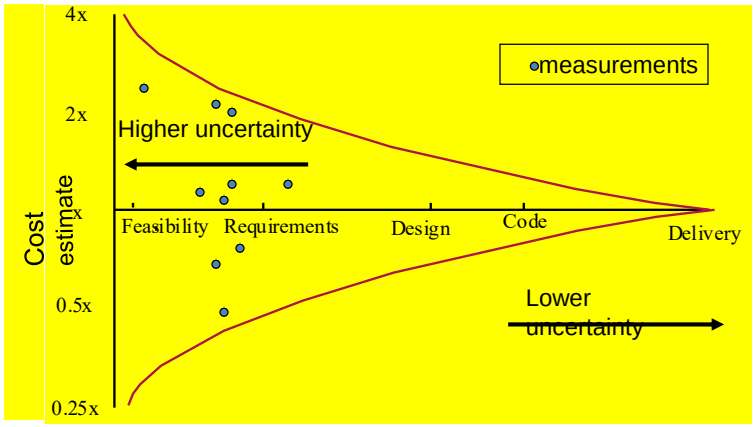
Building Metrics from measurements



New Project estimation using available Metrics



Estimate Uncertainty



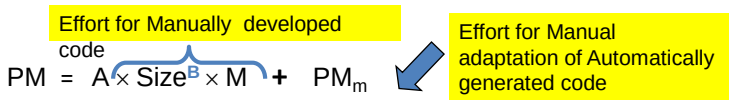
The COCOMO Cost model Constructive Cost Model

- An empirical model based on project experience
- COCOMO'81 is derived from the analysis of 63 software projects in 1981.
- Well-documented, 'independent' model which is not tied to a specific software vendor
- COCOMO II (2000) takes into account different approaches to software development, reuse, etc.

Early Design Level: 7 cost drivers -

COCOMO II

- Estimates can be made after the requirements have been agreed
- Based on standard formula for algorithmic models



- M = PERS × RCPX × RUSE × PDIF × PREX × FCIL × SCED
  - A = 2.5 in initial calibration,
  - Size: manually developed code in **KLOC**
  - Exponent **B**
    - varies from 1.1 to 1.24 depending on novelty of the project, development flexibility, risk management approaches and the process maturity.
    - B is calculated using a Scale Factor based on 5 exponent drivers
  - PM_m: represents manual adaptation for automatically generated code

Exponent scale factors - COCOMO II

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.

Effects of cost drivers

Maximum & Minimum Data are from ref: Boehm, 1997

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

Maximum

Minimum

Effects of cost drivers (M = ?)

Maximum & Minimum Data are from ref: Boehm, 1997

Exponent value	1.17
System size (including factors for reuse and requirements volatility)	128, 000 DSI
Initial COCOMO estimate without cost drivers (M=1)	**730 person-months**
Reliability	Very high, multiplier = 1.39
Complexity	Very high, multiplier = 1.3
Memory constraint	High, multiplier = 1.21
Tool use	Low, multiplier = 1.12
Schedule	Accelerated, multiplier = 1.29
Adjusted COCOMO estimate: $M = \prod (M_i) = 3.15$	**2306 person-months**
Reliability	Very low, multiplier = 0.75
Complexity	Very low, multiplier = 0.75
Memory constraint	None, multiplier = 1
Tool use	Very high, multiplier = 0.72
Schedule	Normal, multiplier = 1
Adjusted COCOMO estimate: $M = \prod (M_i) = 0.405$	**295 person-months**