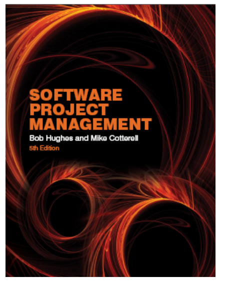
Project Management
Software Effort Estimation

What makes a successful project?

What makes a successful project?

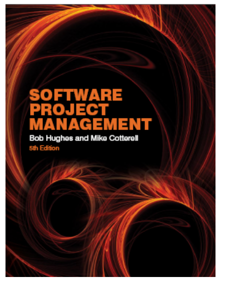


- Delivering:
  - agreed functionality
  - on time
  - at the agreed cost
  - with the required quality
- Stages:
  - 1. set targets
  - 2. Attempt to achieve targets

Problems with estimation

- Subjective nature of much of estimating
  - It may be difficult to produce evidence to support your precise target
- Political pressures
  - Managers may wish to reduce estimated costs in order to win support for acceptance of a project proposal
- Changing technologies
  - these bring uncertainties, especially in the early days when there is a 'learning curve'
- Projects differ
  - Experience on one project may not be applicable to another

Basis for estimation



- Information about past projects
  - Need to collect performance details about past project: how big were they? How much effort/time did they need?
- Need to be able to measure the amount of work involved
  - Traditional size measurement for software is ‘lines of code’ – but this can have problems

Estimation Strategies

These strategies are simple but risky:

Expert judgement	Consult experts and compare estimates *cheap, but unreliable*
Estimation by analogy	Compare with other projects in the same application domain *limited applicability*
Parkinson's Law	Work expands to fill the time available *pessimistic management strategy*
Pricing to win	You do what you can with the budget available *requires trust between parties*

Estimation Techniques

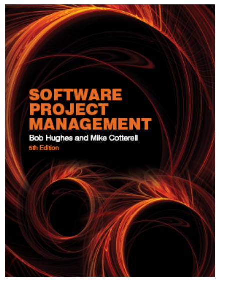
"Decomposition" and "Algorithmic cost modeling" are used together

Decomposition	Estimate costs for components + integration ☞ *top-down or bottom-up estimation*
Algorithmic cost modeling	Exploit database of historical facts to map size on costs ☞ *requires correlation data*

Top-down and bottom-up estimation

- Top-down
  - Start at the system level and assess the overall system functionality and how this is delivered through sub-systems.
  - based on past project data
  - produce overall estimate based on project cost drivers
  - divide overall estimate between jobs to be done
- Bottom-up
  - Start at the component level and estimate the effort required for each component. Add these efforts to reach a final estimate.
  - use when no past project data
  - identify all tasks that have to be done – so quite time-consuming
  - use when you have no data about similar past projects

Bottom-up estimating



1. Break project into smaller and smaller components
2. [Stop when you get to what one person can do in one/two weeks]
3. Estimate costs for the lowest level activities
4. At each higher level calculate estimate by adding estimates for lower levels

Top-down estimate

- Produce overall estimate using effort driver(s)
- distribute proportions of overall estimate to components

Parametric Models

- Examples of parametric models:
  - Albrecht/IFPUG function points
  - Symons/Mark II function points
  - COSMIC function points
  - COCOMO81 and COCOMO II

COCOMO

- Most widely used model for effort and cost estimation.
- Boehm's observations:
  - effort increases faster than the application's size
  - duration increases exponentially with the effort.
  - estimated the parameters for these relationships.
- The COCOMO model distinguishes 3 types of projects:
  - **simple**: small team, familiar environment, familiar type of application
  - **semidetached**: experienced people, unfamiliar environment or new type of application
  - **embedded**: rigid constraints, application embedded in complex hard/software system, rigid requirements, high validation requirements, little experience

COCOMO I formula

- Effort in Person-months = a x KLOC^b
- Duration = c x Effort^d
- Duration = c x (a x KLOC^b)^d

Software Project	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

b exponentiation – 'to the power of…'

adds disproportionately more effort to the larger projects, takes account of bigger management overheads

COCOMO 1 Examples

- Simple project, 32000 lines of code:
  - person months = $2.4 \times (32)^{1.05} = 91$
  - time = $2.5 \times (91)^{0.38} = 14$
  - number of people = $91/14 = 6.5$
- Embedded system, 128000 lines of code:
  - person months = $3.6 \times (128)^{1.2} = 1216$
  - time = $2.5 \times (1216)^{0.32} = 24$
  - number of people = $1216/24 = 51$

COCOMO II formula

- There are different COCOMO II models for estimating at the 'early design' stage and the 'post architecture' stage when the final system is implemented. Look at first one
- The core model is:
  $pm = A(size)^{(sf)} \times (em1) \times (em2) \times (em3) \ldots.$

  where pm = person months, **A** is 2.94, **size** is number of thousands of lines of code, **sf** is the scale factor, and **em** is an effort multiplier

Due to
http://sunset.usc.edu/research/COCOMOII/

Basic COCOMO II cost drivers

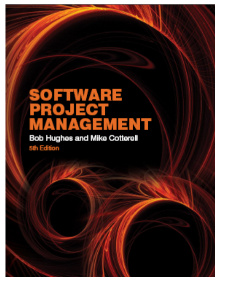
Sym.	Abr.	Name
SF_1	PREC	Precendentedness
SF_2	FLEX	Development Flexibility
SF_3	RESL	Architecture and Risk Resolution
SF_4	TEAM	Team cohesion
SF_5	PMAT	Process Maturity
EM_1	RELY	Required Software
EM_2	DATA	Data Base Size
EM_3	CPLX	Product Complexity
EM_4	RUSE	Required Reusability
EM_5	DOCU	Documentation Match to Life-cycle Needs
EM_6	TIME	Time Constraint
EM_7	STOR	Storage Constraint
EM_8	PVOL	Platform Volatility
EM_9	ACAP	Analyst Capability
EM_{10}	PCAP	Programmer Capability
EM_{11}	AEXP	Applications Experience
EM_{12}	PEXP	Platform Experience
EM_{13}	LTEX	Language and Tool Experience
EM_{14}	PCON	Personnel Continuity
EM_{15}	TOOL	Use of Software Tools
EM_{16}	SITE	Multi-Site Development
EM_{17}	SCED	Required Development Schedule

Due to
http://sunset.usc.edu/research/COCOMOII/

COCOMO II Scale Factors

- Based on five factors which appear to be particularly sensitive to system size
  - Precedentedness (PREC). Degree to which there are past examples that can be consulted
  - Development flexibility (FLEX). Degree of flexibility that exists when implementing the project
  - Architecture/risk resolution (RESL). Degree of uncertainty about requirements
  - Team cohesion (TEAM).
  - Process maturity (PMAT)

Example



- A software development team is developing an application which is very similar to previous ones it has developed.
  - A very precise software engineering document lays down very strict requirements. PREC is very high (score 1.24).
  - FLEX is very low (score 5.07).
  - The good news is that these tight requirements are unlikely to change (RESL is high with a score 2.83).
  - The team is tightly knit (TEAM has high score of 2.19), but processes are informal (so PMAT is low and scores 6.24)

Example calculation

- The formula for sf is

  sf = B + 0.01 × Σ scale factor values

  i.e. sf = 0.91 + 0.01

  × (1.24 + 5.07 + 2.83 + 2.19 + 6.24)

  = 1.0857

- If system contained 10 kloc then estimate would be $2.94 \times 10^{1.0857} = 35.8$ person months
- Using exponentiation ('to the power of') adds disproportionately more to the estimates for larger applications

Effort multipliers

- As well as the scale factor effort multipliers are also assessed:
  - RCPX Product reliability and complexity
  - RUSE Reuse required
  - PDIF Platform difficulty
  - PERS Personnel capability
  - PREX Personnel experience
  - FCIL Facilities available
  - SCED Schedule pressure

Example

- Say that a new project is similar in most characteristics to those that an organization has been dealing for some time
- except
  - the software to be produced is exceptionally complex and will be used in a safety critical system.
  - The software will interface with a new operating system that is currently in beta status.
  - To deal with this the team allocated to the job are regarded as exceptionally good, but do not have a lot of experience on this type of software.

Example continued

RCPX	very high	1.91
PDIF	very high	1.81
PERS	extra high	0.50

All other factors are nominal.

Say estimate is 35.8 person months.

With effort multipliers this becomes 35.8 x 1.91 x 1.81 x 0.5 = 61.9 person months

Conclusions: how to review estimates?

- Ask the following questions about an estimate
  - What are the task size drivers?
  - What productivity rates have been used?
  - Is there an example of a previous project of about the same size?
  - Are there examples of where the productivity rates used have actually been found?