Project Estimation
Kostas Kavoussanakis, EPCC

Overview

Aim:

- To raise awareness of the importance of estimation to project welfare
- To discuss techniques and methods
- To link estimation with the other process activities
Want to start a project?

It is important that you know in advance:
- How long it will take
- How many people it will need
- How much effort it will require

You can then discuss cost
- ...and shake hands

Estimation is hard
- Projects overrun
- Projects go overbudget

---

Two thirds of all projects substantially overrun their estimates;

The average large project misses its delivery date by 25 to 50 percent;

The size of the average schedule slip increases with the size of the project.

Table from B. Boehm
Why so hard?

- Estimates are needed and relied upon **early**
- The functional requirements do not provide a solid background
- It is not immediately known how long it will take to develop the features
  - Particularly if the desired outcomes are genuinely novel.
- **Feature Creep** is a killer
  - It is the unpredictable yet near-certain change of the functionality as the project progresses.
Why so hard? (cont)

- Staff ability
  - Estimators
  - Programmers
- Code reuse
  - Is code reused?
  - Is code to be reused?
- Programming language used

Effects of underestimation

- Understaffing
- Underscoping the quality assurance effort
- Setting too short a schedule.
- These in turn could lead to:
  - staff burnout
  - low quality
  - loss of credibility as deadlines are missed.
Effects of overestimation

- Parkinson's Law:
  
  *Work expands to fill available time*

- The project will take at least as long as estimated even if it was originally overestimated.

When to estimate

- The first estimate is necessary before the start of the project.
- Estimation is a process of gradual refinement.
- It does not finish until the project finishes.
The effort is concentrated where the money is
- MIS
- real-time systems

What is wrong with academic projects?
- Complex computations
- Challenging algorithms

Good estimation practices can keep the project on track and even earn some time for the tricky, interesting areas.

We should standardise the estimation process
- First inside the individual research centres
- Then (quite a big step) across the centres in the UK
Remember the project triangle

- Quality for software projects
  - Functionality
  - Correctness of the code
- Cost and time depend on **Effort** and **Schedule**

**Effort** is the total number of time units (e.g. weeks or months) needed to complete the task.

This may break down to effort from more than one persons, so as to take advantage of certain skills and parallelise the work to gain overall time.

**Caveat:**
- the more people one adds to a project the more one needs to work so as to coordinate them and the more they communicate so as to interact successfully, thus yielding overheads.

**COCOMO:**
- For “large” projects
  \[ \text{Effort} = a \times \text{size}^b \]
- For projects that can be achieved with 2-3 people teams
  \[ \text{Effort} = a \times \text{size} + b \]
Effort and Schedule (cont)

- Effort division may cause gaps in personnel utilisation.
- **Schedule** is the breakdown of effort per person at any given time.
- It is sometimes defined as the total time for the project.
- Schedule can be derived from effort.
- Rule of thumb:
  
  \[
  \text{optimal schedule} = 3 \times \text{effort}^{1/3} \quad (\text{McConnell})
  \]

Estimation Methods

- Expert Opinion
- Estimation by Analogy
- Metrics
Expert Opinion

- The classic method
- Prior experience is the key to estimation
  - This is true in all cases
  - Good practice: only use documented data
- Estimation is left to a senior member of staff, possibly the Project Leader
  - Good practice: involve the developers
- Techniques to mitigate single point of failure
  - Work Break Structure
  - Delphi

Work Break Structure (WBS)

- Two hierarchies:
  - Software product
  - Process
**Product Hierarchy**

- **Software Application**
  - **Component A**
  - **Component B**
  - **Component C**
  - **Sub-component B1**
  - **Sub-component B2**

**Process Hierarchy**

- **Development Activities**
  - **System Engineering**
  - **Programming**
  - **Maintenance**
    - **Detailed Design**
    - **Code and Unit Test**
Work Break Structure (cont)

- Lay out what you want to do
  - Requirements at an early stage
  - Architecture document at a later one
  - Other documents (e.g. priorities list)
- Break the problem into components (workpackages)
- Then break the workpackages in tasks
- Perhaps go another level down
  - Good practice: the more you break and estimate the better
    - “The sum of the errors is less than the error of sums”

Work Break Structure (cont)

- Appreciate the workload of each component
  - Good practice: How many days to the week?
  - Good practice: Check documented experience of similar tasks
Gantt Charts

- Turn back to the workpackages and seek dependencies
  - E.g. “WP1 and WP2 can run in parallel, but can’t start WP3 until WP1 is finished”

- How to create a Gantt chart
  - Get a big piece of paper and loads of post-its
  - Each post-it is a task
  - Try to parallelise tasks as much as dependencies let you
    - horizontal axis represents time, vertical axis resources
  - Length of chart is time
  - Width of chart is effort
    - Gaps represent underutilisation of staff
Participants are asked to make assessment individually
Results are collected, tabulated, and then returned
  – Group discussion optional
Second round of individual assessments taking into account the previous results.

Pros:
  – Applicable to very original projects.
  – Inherent local calibration.
  – WBS can lead to a well documented process.

Cons:
  – Big dependency on experts’ abilities.
  – Big dependency on experts’ presence; how easy does Delphi cope with high staff turnover in a small organisation?
  – There is so much information outside your establishment
The cost of a new project is estimated by analogy to similar completed projects.

Estimates based on historical data from within an organisation are more accurate than estimates based on rules of thumb or educated guesswork.

International Software Benchmarking Standards Group (ISBSG) maintains and exploits a repository of international software project metrics to help software and IT business customers with project estimation, risk analysis, productivity and benchmarking.

Pros:
- Can be accurate.
- Simple if the organisation repeats similar work.
- Estimates are immediately available.
- Encourages detailed documentation.

Cons:
- Can be very unreliable.
- It is very hard to assess the differences between the environments and thus assess the accuracy of the ISBSG data.
No definition for project size
- “Something that encompasses all 3 aspects of a project”
- “The area of the project triangle”

A metric makes estimation:
- Transparent: The abilities of the practitioner are irrelevant
- Repeatable: Exercised by various people at different times yields the same results
- Reliable: The end results, although never accurate, can be closer to the Truth

Additionally:
- The productivity of staff and organisations can be monitored.
- The results can be shared across the globe.

How many lines in your files?

Pros:
- Very natural
- Easily countable

Cons:
- Available too late
- What is a line?
  - Comments?
  - Compare
    
    \[
    d = c++; \\
    \text{with} \\
    c++; \\
    d = c;
    \]

Lines of Code
Function Points

- Function Points provide a more objective and reliable estimate of the size of a project
- “Measure of the size of computer applications and the projects that build them”
- With Function Points one can:
  - Measure Productivity (100 FPs produced this month)
  - Estimate development and support (100 FPs required thus XXX man months)
  - Monitor outsourcing agreements (this library requires 100 FPs, says the contract, these are only XXX)
  - Normalise other measures (100 defects in a 100 FPs is far worse than in 10000 FPs)

Function Points (cont)

- Started in IBM in the late seventies (Alan Albrecht)
- International Function Point Users Group founded in late eighties
- An active area of research (COSMIC-FFP)
- Measurements are taken from the user's point of view
  - Independent of computer language, development methodology, technology or capability of the project team

http://www.ifpug.org/home/docs/ifpughome.html
Function Points Evaluation

Pros:
- Prescriptive method for sizing.
- Can be applied reasonably early in project lifetime.
- Immune to language and platform idiosyncrasies.
- Large user base-active effort.

Cons:
- Manual, fiddly process.
- Disagreement about its applicability across the various types of modern projects.
- Not ideal in the requirements capture period; although the method can be applied, it is too much work for such a volatile description.

Estimation Tools

Approaches depending on estimation method:
- Expert Judgement
- Algorithmic Models
  - FPs, for size
  - COCOMO, or SLIM, or tables based on size estimation for effort
  - SLIM or tables based on size of effort estimation for schedule

Most products are hybrids:
- Organising expert judgement
- Exploiting machine learning
- Refining algorithmic models
With estimation tools you can:

- Estimate size using Function Points or other metrics
- Derive effort and schedule using various algorithms and techniques
- Perform “what if” analyses with staffing, duration etc. and appreciate how realistic they are
- Produce and update Gantt and other charts easily
- Maintain and exploit a database of historic data
- Import data from other projects run in organisations with which you have no connection

Tools are important for estimation

- They make estimation feel like an algorithm
- They prevent from skipping necessary tasks
- They help organise, update and archive the results

Additionally, they provoke you to think about the lifetime of the project and at the same time do the dirty job for you

Make your own research for tools and their efficiency

- Compare them against already finished projects
- Use them in parallel with your current estimation technique
Estimation is a part of Process

- The most important issue about estimation is frame of mind
- It takes more than a good estimate to keep a project in good shape
- Important things to note:
  - The silver bullet syndrome
  - Choice of development model
  - Track the project progress
  - Estimate in ranges
  - Think before you quote
  - Bring in the right people

The silver bullet syndrome

- None of the estimation methods is perfect
- None of the estimation tools is perfect
- Evaluate and calibrate methods and tools
Development model

- Many models are flexible enough to accommodate changes.
- Design to schedule is great for fixed time-fixed cost projects.
- Iterative models are better for projects with many unknowns.
  - Staged delivery
  - Design to schedule
  - Evolutionary delivery
  - Spiral

Keep track of your project

- A software project is a live entity with complex behaviour
- Monitor and update estimates
  - Even after the design stage you can expect to be off by 25%
- Complement your estimates with a good tracking policy
  - Various tools available
- Always update the risks list and the priorities list.
Estimate in ranges

- The tools and methods may give you absolute numbers
- Add some padding, i.e. slack time to make up for errors
  - Do it and the customer does not trust your estimate
  - Don’t do it (or don’t pad enough) and you may overrun
  - Good practice: estimation range
  - Good practice: buffers for specific risks
- Risk analyses will show things that can go wrong
- Evaluate their impact to the schedule
- Provide conditional estimates
  - Start with raw numbers (from tools and methods)
  - Consider risks
  - Consider what could help

Example:
- “The GUI will take 6 months, +2 if the tool-generated code is useless, +1 if Jo goes on holiday, -1 if we can reuse the ‘File’ menu functionality from project X.
  6+3-1”.
- Alternatively:
  [5-9]
- Very different from “8”
  - 6+2+1-1=8.
- Incorporation of risks to estimate
Think before you quote

- When tempted to provide an estimate thinking on your feet...
  
  **DON'T!**

- People remember it, pass it on and hold you accountable.
  - Although it is normal for the estimate at the feasibility stage to be off by 400%.

Bring in the right people

- 2 groups of colleagues one cannot ignore.
- Group 1: those experienced in estimation.
  - They know how to appreciate the workload of components
    - Trust them in conjunction with documented past estimates
  - They know how to do it.

- Group 2: those who will do the work
  - They will know how long it will take them
    - Do not judge the effort from your own standards
  - They can see things from a different angle
    - And foresee the technical challenge
  - They get a feeling of ownership
    - Persistence and application to the project
References


References (cont)

References (cont)


References (cont)

References (cont)