Software Estimation

Some content adapted from “Rapid Development” by Steve McConnell

Project Management

- Two over-arching inter-dependent aspects of software projects
  - Process
  - Project Management

Project Management

- Main responsibilities of a project manager are
  - Project planning
  - Project monitoring and control

(Major activities: Software Estimation, Scheduling and Tracking)

Software Estimation

- Predictions are hard, especially about the future
- Two Types of estimates:
  - Lucky or Lousy

Estimations

- Created, used or refined during
  - Strategic planning
  - Feasibility study
  - Proposals
  - Vendor and sub-contractor evaluation
  - Project planning (iteratively)
- Basic process
  1) Estimate the **size** of the product
  2) Estimate the **effort** (man-hours/man-months)
  3) Estimate the **schedule**
  - NOTE: Not all of these steps are always explicitly performed

Estimation Activity

Week 3 - SSAD

Estimate how long will it take for you to get to hostel from class today

Estimations

- Remember, an “exact estimate” is an oxymoron

- On what basis did you estimate?
  - Experience right? (History matters…)
  - Likely as an “average” probability
  - For most software projects there is no such ‘average’

 # 1: Always give a range
Never give them a number

Numbers are for facts;
Ranges are for estimates;

Estimation exercise

- One result per table
- Use one of three collaboration techniques based on the allocated groups
- GROUP 1
  - Estimate as a group, come to consensus
- GROUP 2
  - Divide the work among you
- GROUP 3
  - First estimate individually
  - Then combine the estimates as a group

Estimation exercise 1 (10-12 min)

1. Surface temperature of the sun (in degrees C)
2. Latitude of Shanghai (in degrees)
3. Surface area of Asia (in km^2)
4. Birth date of Alexander The Great (year)
5. Global revenue of "Titanic" (in $)
6. Length of the Pacific coastline (Ca, Or, Wa) (in km)
7. Number of books published in USA, 1776 to 2004
8. Weight of the largest whale (in tonnes)

This is adapted from "Software Estimation" by Steve McConnell

On a side note…

An engineer, a mathematician and an accountant are sitting at the bar

The barman asks: “What’s 68+73 ?”

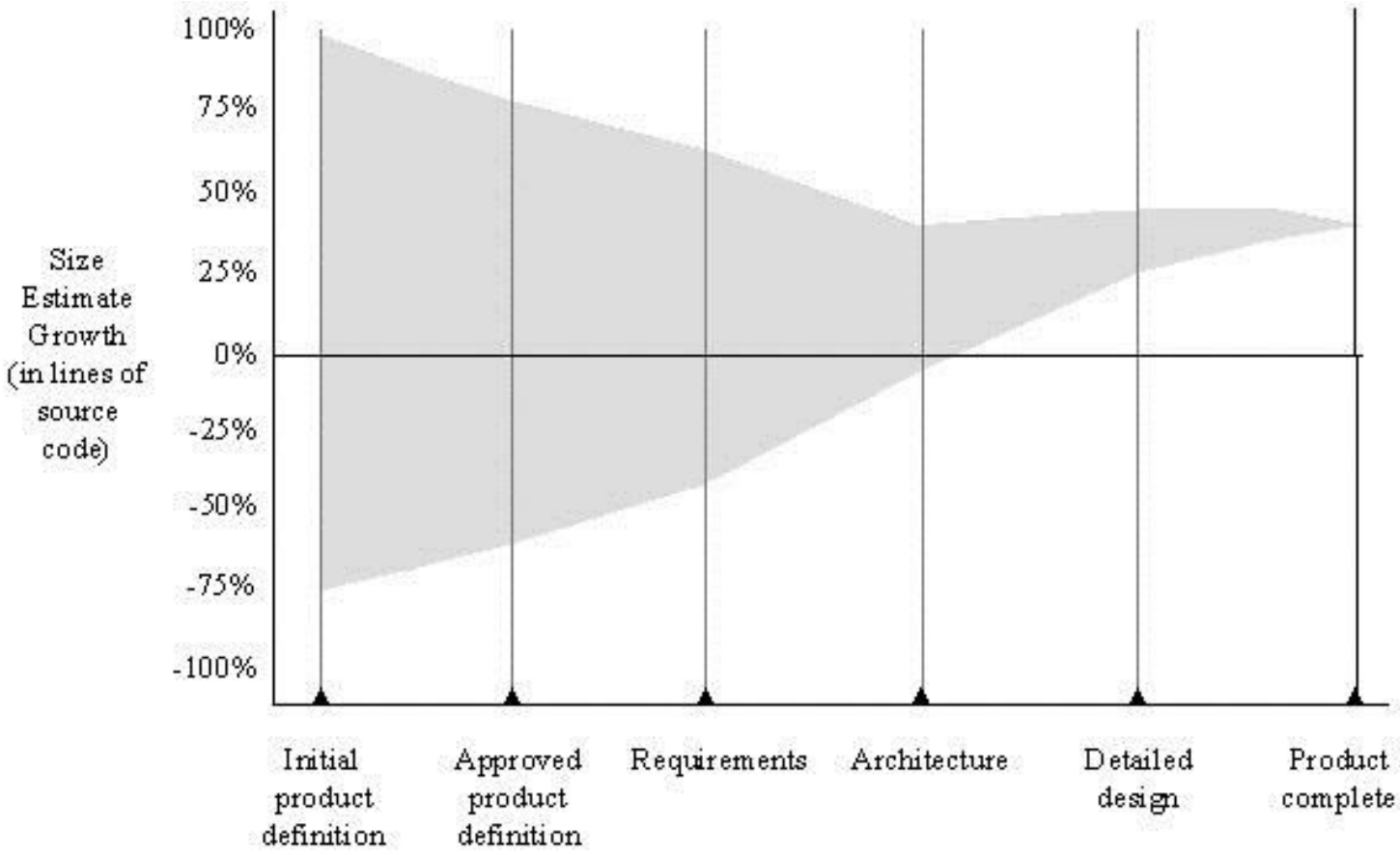
- Engineer: 141
- Mathematician: 68 + 73 = 73 + 68
- Accountant: Usually it’s 141, but what do you want to do with the number?

#2 Always ask what the estimate will be used for

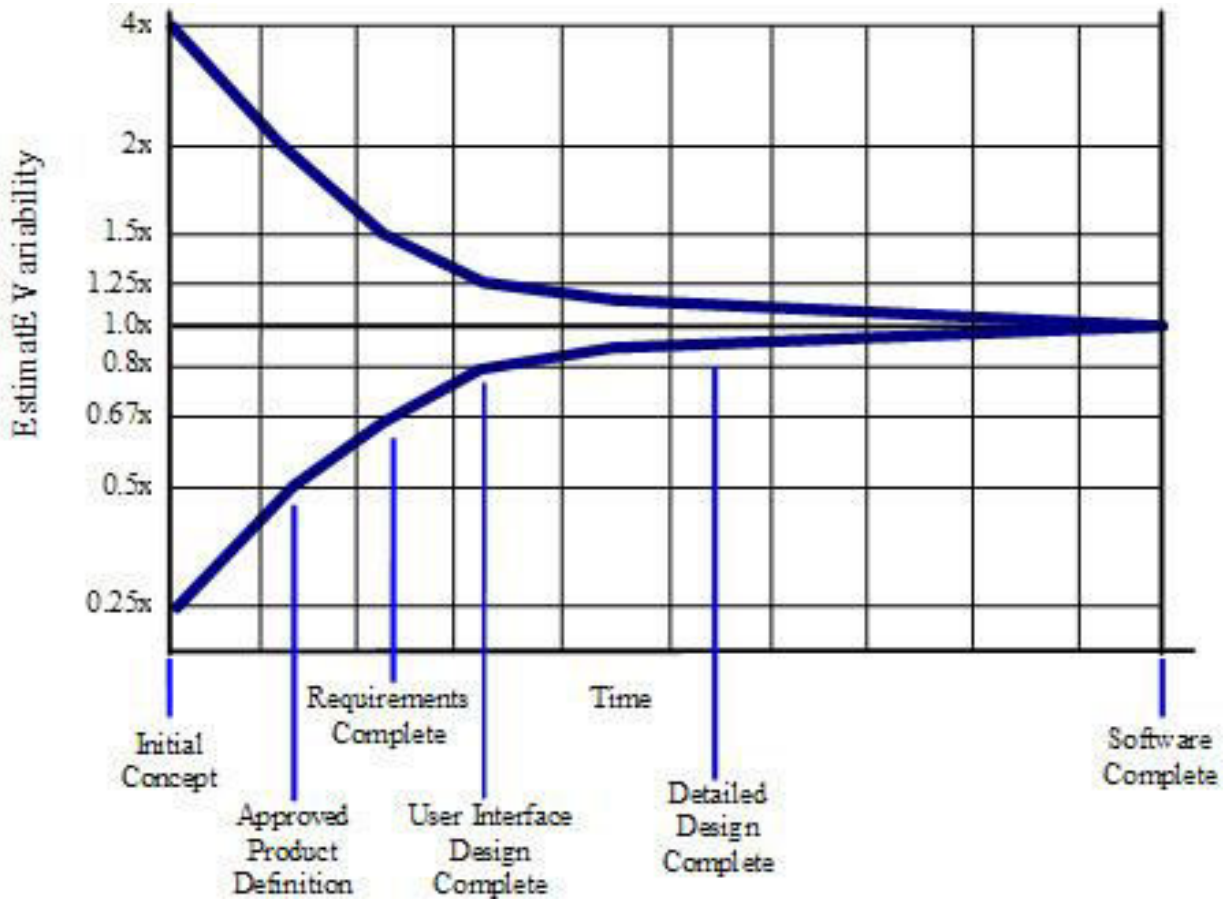
#3 Estimation != Commitment

Getting an estimate wrong doesn’t hurt

Cone of Uncertainty



Cone of Uncertainty



From Rapid Development by Steve McConnell

Estimation

- Target vs. Committed Dates
  - Target: Proposed by business or marketing
  - Do not commit to this too soon!
  - Committed dates: Team agrees to this

  – Let's looks an assignment analogy
    - Do instructors take the various factors into consideration before assigning a deadline?

What can you count?

- Number of stakeholders
- Number of goals
- Number of events
- Number of business processes
- Number of high-level user stories
- Number of detailed user stories
- Number of screens
- ....

#4 First try to measure, count and compute

Estimate only when necessary

#5 Aggregate independent estimates

"Wisdom of the Crowds"

The law of large numbers
(or: statistics is on our side, for once)

- If we estimate with an error of x%
- The estimate of each scope item will have an error of x%
- But...
- Some items will be over-estimated, others under-estimated (maybe....)
- => The error on the total estimate is < x%

Answers

- Sun: 6000° C
- Shanghai: 31 degrees North
- Asian area: 44,390,000 km²
- Alexander was born in 356 BC
- Titanic: 1.835 billion $
- Pacific Coast: 1293 kilometres
- Published books: 22 million
- Whale: 170 tonnes

This quiz is from “Software Estimation” by Steve McConnell (Microsoft Press)

And the winner is?

Estimation Methodologies

- Top-down
- Bottom-up
- Analogy
- Expert Judgment
- Priced to Win (request for quote – RFQ)
- Parametric or Algorithmic Method
  - Using formulas and equations

Wideband Delphi

- Group consensus approach
- Present experts with a problem and response form
- Conduct group discussion, collect anonymous opinions, then feedback
- Conduct another discussion & iterate until consensus
- Advantages
  - Easy, inexpensive, utilizes expertise of several people
  - Does not require historical data
- Disadvantages
  - Difficult to repeat
  - May fail to reach consensus, reach wrong one, or all may have same bias

Function Points

- Software size measured by number & complexity of functions it performs
- More methodical than LOC counts
- House analogy
  - House's Square Feet ~= Software LOC
  - # Bedrooms & Baths ~= Function points
  - Former is size only, latter is size & function
- Six basic steps

Function Point Process

- 1. Count # of business functions per category
  - Categories: outputs, inputs, DB inquiries, files or data structures, and interfaces
- 2. Establish Complexity Factor for each and apply
  - Low, Medium, High
  - Set a weighting multiplier for each (0 → 15)
  - This results in the "unadjusted function-point total"
- 3. Compute an "influence multiplier" and apply
  - It ranges from 0.65 to 1.35; is based on 14 factors
- 4. Results in "function point total"
  - This can be used in comparative estimates

Function point multipliers

	Function Points		
Program Characteristic	Low Complexity	Medium Complexity	High Complexity
Number of Inputs	x 3	x 4	x 6
Number of Outputs	x 4	x 5	x 7
Inquiries	x 3	x 4	x 6
Logical internal files	x 7	x 10	x 15
External interface files	x 5	x 7	x 10

From Rapid Development by Steve McConnell

Counting the Number of Function Points

	Function Points		
Program Characteristic	Low Complexity	Medium Complexity	High Complexity
Number of Inputs	5 x 3 = 15	2 x 4 = 8	3 x 6 =18
Number of Outputs	6 x 4 = 24	6 x 5 = 30	0 x 7 = 0
Inquiries	0 x 3 = 0	2 x 4 = 8	4 x 6 = 24
Logical internal files	5 x 7 = 35	2 x 10 =20	3 x 15 = 45
External interface files	8 x 5 = 40	0 x 7 = 0	2 x 10 = 20
Unadjusted function-point total			287
Influence multiplier			1.20
Adjusted function-point total			344

From Rapid Development by Steve McConnell

Code Reuse & Estimation

- Does not come for free
- Code types: New, Modified, Reused
- If code is more than 50% modified, it's "new"
- Reuse factors have wide range
  - Reused code takes 30% effort of new
  - Modified is 60% of new
- Integration effort with reused code almost as expensive as with new code

Effort Estimation

- Now that you know the “size”, determine the “effort” needed to build it
- Various models: empirical, mathematical, subjective
- Expressed in units of duration
  - Man-months (‘staff-months’) or Man-hours

COCOMO

- Barry Boehm – 1980's
- **CO**nstructive **CO**st **MO**del
- Input – LOC, Output - Person Months
- Allows for the type of application, size, and "Cost Drivers"
- Cost drivers using High/Med/Low & include
  - Motivation, Ability of team, Application experience, etc.
- Biggest weakness?
  - Requires input of a product size estimate in LOC

Estimation Issues

- Quality estimations needed early but information is limited
- Precise estimation data available at end but not needed
  - Or is it? What about the next project?
- Best estimates are based on past experience
- Politics of estimation:
  - You may anticipate a "cut" by upper management
- For many software projects there is little or none
  - Technologies change
  - Historical data unavailable
  - Wide variance in project experiences/types
  - Subjective nature of software estimation

Over and Under Estimation

- Over estimation issues
  - The project will not be funded
    - Conservative estimates guaranteeing 100% success may mean funding probability of zero.
  - Danger of feature and scope creep
  - Be aware of "double-padding": team member + manager

- Under estimation issues
  - Quality issues (short changing key phases like testing)
  - Inability to meet deadlines
  - Morale and other team motivation issues

Estimation Guidelines

- Estimate iteratively!
  - Process of gradual refinement
  - Make your best estimates at each planning stage
  - Refine estimates and adjust plans iteratively
  - Plans and decisions can be refined in response
  - Balance: too many revisions vs. too few

Know Your Deadlines

- Are they ‘Real Deadlines’?
  - Tied to an external event
  - Have to be met for project to be a success
  - Ex: end of financial year, contractual deadline, Y2K
- Or ‘Artificial Deadlines’?
  - Set by arbitrary authority
  - May have some flexibility (if pushed)

Estimation "Presentation"

- How you present the estimation can have **huge** impact
- Techniques
  - Plus-or-minus qualifiers
    - 6 months +/-1 month
  - Ranges
    - 6-8 months
  - Risk Quantification
    - +/- with added information
    - +1 month of new tools not working as expected
    - -2 weeks for less delay in hiring new developers
  - Cases
    - Best / Planned / Current / Worst cases
  - Coarse Dates
    - Q3 02
  - Confidence Factors
    - April 1 – 10% probability, July 1 – 50%, etc.

Other Estimation Factors

- Account for resource experience or skill
  - Up to a point
  - Often needed more on the "low" end, such as for a new or junior person
- Allow for "non-project" time & common tasks
  - Meetings, phone calls, web surfing, sick days
- There are commercial 'estimation tools' available
  - They typically require configuration based on past data

Summary

- Software estimation involves estimation of
  - Size
  - Effort
  - Resources
- There are various estimation techniques. For example
  - Wideband Delphi
  - CoCoMo