Estimation

Adapted from John Musser
Estimation

• “Predictions are hard, especially about the future”, Yogi Berra
• 2 Types: Lucky or Lousy?
Planning, Estimating, Scheduling

- What’s the difference?
- Plan: Identify activities. No specific start and end dates.
- Estimating: Determining the size & duration of activities.
- Schedule: Adds specific start and end dates, relationships, and resources.
Project Planning: A 12 Step Program

- Set goal and scope
- Select lifecycle
- Set org./team form
- Start team selection
- Determine risks
- Create WBS

- Identify tasks
- Estimate size
- Estimate effort
- Identify task dependencies
- Assign resources
- Schedule work
How To Schedule

• 1. Identify “what” needs to be done
  – Work Breakdown Structure (WBS)
• 2. Identify “how much” (the size)
  – Size estimation techniques
• 3. Identify the dependency between tasks
  – Dependency graph, network diagram
• 4. Estimate total duration of the work to be done
  – The actual schedule
Partitioning Your Project

- You need to decompose your project into manageable chunks
- ALL projects need this step
- Divide & Conquer
- Two main causes of project failure
  - Forgetting something critical
  - Ballpark estimates become targets
- How does partitioning help this?
Project Elements

- A Project: functions, activities, tasks
Estimations

• Very difficult to do, but needed often
• Created, used or refined during
  – Strategic planning
  – Feasibility study and/or SOW
  – Proposals
  – Vendor and sub-contractor evaluation
  – Project planning (iteratively)
• Basic process
  – Estimate the **size** of the product
  – Estimate the **effort** (man-months)
  – Estimate the **schedule**
  – NOTE: Not all of these steps are always explicitly performed
Estimations

• Remember, an “exact estimate” is an oxymoron
• Estimate how long will it take you to get home from class tonight
  – On what basis did you do that?
  – Experience right?
  – Likely as an “average” probability
  – For most software projects there is no such ‘average’
• Most software estimations are off by 25-100%
Estimation

• Target vs. Committed Dates
  • Target: Proposed by business or marketing
  • Do not commit to this too soon!
  • Committed: Team agrees to this
  • After you’ve developed a schedule
Estimation

• Size:
  – Small projects (10-99 FPs), variance of 7% from post-requirements estimates
  – Medium (100-999 FPs), 22% variance
  – Large (1000-9999 FPs) 38% variance
  – Very large (> 10K FPs) 51% variance
Estimation Methodologies

- Top-down
- Bottom-up
- Analogy
- Expert Judgment
- Priced to Win
- Parametric or Algorithmic Method
  - Using formulas and equations
Top-down Estimation

• Based on overall characteristics of project
  – Some of the others can be “types” of top-down
    (Analogy, Expert Judgment, and Algorithmic methods)

• Advantages
  – Easy to calculate
  – Effective early on (like initial cost estimates)

• Disadvantages
  – Some models are questionable or may not fit
  – Less accurate because it doesn’t look at details
Bottom-up Estimation

- Create WBS
- Add from the bottom-up
- Advantages
  - Works well if activities well understood
- Disadvantages
  - Specific activities not always known
  - More time consuming
Expert Judgment

• Use somebody who has recent experience on a similar project
• You get a “guesstimate”
• Accuracy depends on their ‘real’ expertise
• Comparable application(s) must be accurately chosen
  – Systematic
• Can use a weighted-average of opinions
Estimation by Analogy

- Use past project
  - Must be sufficiently similar (technology, type, organization)
  - Find comparable attributes (ex: # of inputs/outputs)
  - Can create a function

- Advantages
  - Based on actual historical data

- Disadvantages
  - Difficulty ‘matching’ project types
  - Prior data may have been mis-measured
  - How to measure differences – no two exactly same
Priced to Win

• Just follow other estimates
• Save on doing full estimate
• Needs information on other estimates (or prices)
• Purchaser must closely watch trade-offs
• Priced to lose?
Algorithmic Measures

- Lines of Code (LOC)
- Function points
- Feature points or object points
- Other possible
  - Number of bubbles on a DFD
  - Number of ERD entities
  - Number of processes on a structure chart
- LOC and function points most common
  - (of the algorithmic approaches)
- Majority of projects use none of the above
Code-based Estimates

• **LOC Advantages**
  – Commonly understood metric
  – Permits specific comparison
  – Actuals easily measured

• **LOC Disadvantages**
  – Difficult to estimate early in cycle
  – Counts vary by language
  – Many costs not considered (ex: requirements)
  – Programmers may be rewarded based on this
    • Can use: # defects/# LOC
  – Code generators produce excess code
LOC Estimate Issues

• How do you know how many in advance?
• What about different languages?
• What about programmer style?
• Stat: avg. programmer productivity: 3,000 LOC/yr
• Most algorithmic approaches are more effective after requirements (or have to be after)
Function Points

• Software size s/b 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 biz functions per category
   - Categories: outputs, inputs, db inquiries, files or data structures, and interfaces

2. Establish Complexity Factor for each and apply
   - Simple, Average, Complex
   - 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
Wideband Delphi

• Group consensus approach
• Rand corp. used orig. Delphi approach to predict future technologies
• 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
Parametric Method Issues

• Remember: most projects you’ll run into don’t use these
• Which is ‘normal’, so don’t be surprised
  – Or come-in to new job and say “Hey, let’s use COCOMO”
• These are more effective on large projects
  – Where a past historical base exists
• Primary issue for most projects are
  – Lack of similar projects
    • Thus lack of comparable data
• Catch-22: how to get started
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 (or ‘staff-months’ now)
Effort Estimation

- McConnell shows schedule tables for conversion of size to effort
- As with parametric size estimation, these techniques perform better with historical data
- Again, not seen in ‘average’ projects
- Often the size and effort estimation steps are combined (not that this is recommended, but is what often is done)
- “Commitment-Based” Scheduling is what is often done
  - Ask developer to ‘commit’ to an estimate (his or her own)
COCOMO

- COnstructive COst MOdel
- Allows for the type of application, size, and “Cost Drivers”
- Outputs in Person Months
- Cost drivers using High/Med/Low & include
  - Motivation
  - Ability of team
  - Application experience
- 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.
  - Parkinson’s Law: Work expands to take the time allowed
  - 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
Other Estimation Notes

• Remember: “manage expectations”
• Parkinson’s Law
  – “Work expands to fill the time available”
• The Student Syndrome
  – Procrastination until the last minute (cram)