Navigating the Minefield: Successful Software Estimation before Requirements are Complete

PRESENTED BY
CAROL A. DEKKERS
CAROLDEKKERS@GMAIL.COM
DANIEL B. FRENCH
DFRENCH@COBEC.COM
Agenda

- Software Project Cost Estimating
- Project requirements - Demystified
- Estimates before Requirements
- DYB and gu(estimate)
- Q & A
Project estimating

Scope & Quality

Effort

Cost

Schedule
**estimate**

[v. es-tuh-meyt; n. es-tuh-mit, -meyt]

- Spell
- Syllables

**Synonyms**

**Examples**

**Word Origin**

**verb (used with object), estimated, estimating.**

1. to form an approximate judgment or opinion regarding the worth, amount, size, weight, etc., of; calculate approximately:
   
   *to estimate the cost of a college education.*

2. to form an opinion of; judge.

**verb (used without object), estimated, estimating.**

3. to make an estimate.

**noun**

4. an approximate judgment or calculation, as of the value, amount, time, size, or weight of something.

5. a judgment or opinion, as of the qualities of a person or thing.

6. a statement of the approximate charge for work to be done, submitted by a person or business firm ready to undertake the work.
Project goals → Faster + Better + Cheaper...
delivered on-time & on-budget!
Typical estimating challenges

- Complete
- Accurate
- Applicable
- Available
- Consistent
- What is (in / out) of scope

INPUTS (KNOWN + ASSUMPTIONS)

ESTIMATING MODEL(S)

ESTIMATES (COST, EFFORT, DURATION, …)
“Dog chasing its tail”
Cycle

Poor requirements → Poor estimates → Poor change management → $ and schedule overruns
Software requirements
Three distinct types of software requirements

1. Functional requirements
2. Non-functional requirements
3. Technical (build) requirements

"User requirements"
"Project requirements"
"Developer construction requirements"
How customers view requirements – non-transparent

- **Functional**
- **Non-functional**
- **Technical**
- **Project requirements**
How developers view requirements – blended

- Functional
- Non-functional
- Technical
- Project requirements
1. Functional requirements

**WHAT software shall do**

- Documented with use cases or user stories

- Responsibility of customers/users to define
- Objective, documented
- Can be quantified (and estimated)
  - Functional size of software (function points: FP)
  - Key input to estimating
- “Floor plan for software”
2. Non-functional requirements

HOW software shall perform: Suitability, Accuracy, Interoperability, Compliance, Security, Reliability, Efficiency, Maintainability, Portability, Quality and “ilities”

- Responsibility of customers/users to define
- Documented with supplementary specs or user stories
- Can be quantified:
  - SNAP points, VAF, COCOMO II effort multipliers
  - Can DOUBLE an estimate
- “International code for construction”
3. Technical requirements

How team will develop or build the software

- May be influenced by Functional and Non-functional requirements plus environment
- Tools, methods, skills, programming language, etc.
- IT responsibility
- “Blueprints for software”
- **NOT** part of Use Cases, not part of FP, but….directly a part of estimate
Segue: Agile User Stories

- Collection of high level software requirements:
  - Functional user stories (functions);
  - Non-functional (how, quality, usability, performance, etc.);
  - May include technical user stories;
  - May include other requirements (such as research spikes or bugs)

- May be inter-related and overlapping
- Details emerge during sprints
- Agile is part of technical requirement (How to build software)
Estimating techniques

- Analogy
- Parametric
- Expert opinion
- Extrapolation from actuals
- Engineering build-up
- Non-trivial
- History is repeatable

Estimates (cost, effort, schedule)
Good estimates build on knowns
Predictive modeling
Examples of estimating formulas

<table>
<thead>
<tr>
<th>Metric</th>
<th>Units</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Cost Ratio (completed projects)</td>
<td>$ / FP (or SLOC)</td>
<td>(Total Hours * Hourly Cost) + Other Costs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Project Functional Size</td>
</tr>
<tr>
<td>Support Cost Ratio</td>
<td>$ / 1000 FP (or FTE / app)</td>
<td>(Support Hours * Hourly Cost) + Other Costs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Application Functional Size</td>
</tr>
<tr>
<td>Repair Cost Ratio</td>
<td>$ / FP (or per fix)</td>
<td>(Repair hours * Hourly Cost)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Functional Size of Repair</td>
</tr>
</tbody>
</table>
Today, estimating with good requirements and a solid model is similar to having a mapped minefield...
Estimating before requirements
Can you estimate an idea?
I have an idea... can you give me a quick estimate?

I won’t hold you to it... “ballpark” cost...
Kind-of-like xxx...
I need some #’s...
today
“Right now it’s just an idea...”
The Project Process Model

Quality
- Expressed
- Expected
- Elusive
- Exciting

Project Measures
- Delivery Rate
- Defect Detection Ratio
- Duration Delivery Rate
- Cost per Unit of Measure

Input
- Time
- Cost

Process
- Attributes
  - People
  - Environment
  - Tools
  - Techniques

Output
- Product Size
- Scrap

Schedules

Deliverables

Define | Design | Construct | Test | Install | Maintain

Resources

An idea happens way before solid data is available
The Cone Of Uncertainty

Estimate with historical data statistics and strive to mitigate the underestimation risk.

Underestimate (Hofstadter's Law)

"Parkinson Squeeze" the current underestimate

@ http://blog.karmona.com
When faced with a napkin, what can you do?

- Refuse to play
- Memory game ("Kinda-sorta")
- Mutter ("professional ethics") and leave
- SWAG ($1 million & 6 months)
- Promise (and disappear)

Or...
DYB  (do your best)

and (gu)estimate
DYB and (gu)estimate

1. Document base assumptions:
   - Functional size
   - Non-functional
   - Technical
   - Historical growth %

2. Document assumptions & do at least two estimates:
   - ROM based on analogy, or expert opinion

3. Save the napkin
1. Document base assumptions

Functional size: (example shortcuts)

- One File Model (# entities * 31 FP each)
- Simple Function Points
- T-shirt sizing (XS - 3X)
- ISBSG (International Software Benchmarking Standards Group) function point profiles
- Patterns (sizing by analogy)
Non-functional size: ("ilities")

- Document assumptions: standard constraints (e.g., performance, security, special needs)
- Consider adjustment factor or software non-functional assessment (SNAP)
- Overestimate the complexity (seldom less complex) & compare history
1. Document base assumptions

Technical requirements:

- standard development tool suite for subject area
- tools used on similar systems
- consider “uniqueness” potential (architecture, method, novel subject matter, special skills sets, etc)
1. Document base assumptions

Adjust for uncertainty based on historical %:

- Size growth average 1.5-2%* per month (18-month project → +30% FP)
- Rework 40% → can double expert opinion
- Risk management → document areas of greatest uncertainty

2. Rough order of magnitude (ROM) equations

- Range of “(gu)estimates” → NO decimals!
- Rules of thumb, high level size
- Supplement with expert opinion (adjusted for uncertainty)
- Label results as “preliminary”
- Save copies of modeling exercises
3. Save the napkin

- Despite best intent → estimators cannot predict future based on ideas
- (Gu)estimates based on documented assumptions are a step forward
- If (gu)estimate becomes project budget/schedule → scan the napkin for Kick-off meeting

Project ABC
Budget $500 K
Preliminary requirements
Garbage In, Garbage Out

YOUR ANALYSIS IS ONLY AS GOOD AS YOUR DATA

// (garbage) = garbage
Now you can carefully... navigate the minefield
QUESTIONS?

... THANK YOU
Navigating the Minefield: Successful Software Estimation *before* Requirements are Complete

PRESENTED BY
CAROL A. DEKKERS
EMAIL: CAROLDEKKERS@GMAIL.COM

DANIEL B. FRENCH
EMAIL: DFRENCH@COBEC.COM