HOW LONG WILL IT TAKE TO FIX THE BUGS IN OUR CONTROL MANAGEMENT SOFTWARE?
Dilbert
Scott Adams

How long will it take to fix the bugs in our control management software?

Do you want a realistic estimate that will ruin your day, or a lie that will allow your ignorance and your happiness to lock arms and square dance to the next cubicle?
How long will it take to fix the bugs in our control management software?

Do you want a realistic estimate that will ruin your day, or a lie that will allow your ignorance and your happiness to lock arms and square dance to the next cubicle?

That second option sounds festive.

I'm a pleaser.
Chapter 4

Where Does Estimation Error Come From?

“There’s no point in being exact about something if you don’t even know what you’re talking about.”

John Von Neumann

Four generic sources of estimation error:

1. Inaccurate information about the project being estimated
2. Inaccurate information about the capabilities of the organization that are needed
3. Too much chaos in the project to support accurate estimation
4. Inaccuracies arising from the estimation process itself
Sources of Estimation Uncertainty

• Software development – a process of gradual refinement
• Start with a general concept
• Refine the concept based on product & project goals
  Budget and schedule needed for given functionality
  Functionality delivered with given budget and schedule
  Flexibility in budget, schedule and functionality

Good example:
  Telephone Number Checker for an order entry system
Telephone Number Checker

**Uncertainties:**

- Check whether the numbers entered are valid?
- Cheap or expensive version?
- Implement a cheap version and then want the expensive version?
- Off the shelf version or develop you own?
- Design alternatives (“at least 10 different versions)?
- How long will it take to code it?
- Should the Telephone Number Checker and the Address Checker interact?
- Quality level of the Telephone Number Checker?
- How long to debug and correct mistakes in implementation?
Cone of Uncertainty

- SW development involves 1,000’s of decisions
- Uncertainty relates to how the decisions will be resolved (variability in scope – size of effort, cost, features, or combination of size, cost and features)
- As more decisions are made, uncertainty is reduced
Cloud of Uncertainty remains

- No focus on reducing variability
- Project not well estimated...
Cone does not narrow itself

Cone narrows only as decisions are made to eliminate variability

• Defining requirements (what is and is not included)
• Defining the user interface reduces risk of misunderstood requirements
Estimation Error by SW Development Activity

- Single point estimate and range without adjusting for uncertainty…. 
- Ranges are therefore too narrow.
- Start with “most likely” estimates then compute ranges using predefined multipliers

<table>
<thead>
<tr>
<th>Phase</th>
<th>Possible Error on Low Side</th>
<th>Possible Error on High Side</th>
<th>Range of High to Low Estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Concept</td>
<td>0.25 × (-75%)</td>
<td>4.0 × (300%)</td>
<td>16 ×</td>
</tr>
<tr>
<td>Approved Product Definition</td>
<td>0.50 × (-50%)</td>
<td>2.0 × (100%)</td>
<td>4 ×</td>
</tr>
<tr>
<td>Requirements Complete</td>
<td>0.67 × (-33%)</td>
<td>1.5 × (50%)</td>
<td>2.25 ×</td>
</tr>
<tr>
<td>User Interface Design Complete</td>
<td>0.80 × (-20%)</td>
<td>1.25 × (25%)</td>
<td>1.6 ×</td>
</tr>
<tr>
<td>Detailed Design Complete</td>
<td>0.90 × (-10%)</td>
<td>1.10 × (10%)</td>
<td>1.2 ×</td>
</tr>
</tbody>
</table>

(sequential projects)

Source: Adapted from *Software Estimation with Cocomo II* (Boehm et al. 2000)
Commitment

- Making commitments too early in the “Cone”
- *Competent* Managers can complete the project if the estimate at the beginning turns out to be within 20% of the actual time it takes
- If the estimate is actually off by “several hundred percent”, getting to completion is not going to happen!
- The advice:
  Delay commitments until uncertainty has been reduced (you reach the narrow part of the cone)… maybe 30% of the way into the project.
How does this work for *iterative* projects?

- This means a full development cycle for each iteration
- You go through a miniature Cone in each iteration
- Before requirements work, the Approved Product Definition work is subject to \(4\) variability from high to low estimates.

**Short iterations:**

- You can move from Approved Product Definition to Req’ts Complete to User Interface Design Complete in a few days
- Variability goes from \(4\) to \(1.6\) for the specific features delivered
- Lose long-run predictability
- Many teams define most requirements and user interface design done upfront with design, construction, test and release done in iterations
- Result is that variability is about \(25\%\)
Chaotic Development

Adds variability!

**Examples:**

- Req’ts not investigated initially
- Little end-user involvement in req’ts validation
- Poor designs that result in bugs
- Poor coding that requires major bug fixing
- Inexperience developers

- Incomplete or unskilled project planning
- Prima donna team members
- Abandoning planning under pressure
- Developer gold-plating
- Lack of automated source code control
Unstable Requirements

- *Cone of Uncertainty* cannot be narrowed
- Well-run projects, establish an initial set of requirements that are “baselined”
- Any proposed changes should be analyzed... impacts understood and tradeoffs made
- Negotiated changes are reflected in modifications in schedule and costs
Estimating Requirements Growth

The effect of “unstable” requirements

- Allow for requirements growth in the estimates
- Figure 4-5
  Accounts for 50% growth over the course of the project
Omitted Activities – what is missed

**Functional** Req’ts:
- Setup/installation program
- Data conversion utility
- Glue code needed to use 3rd party or open-source SW
- Help system
- Deployment modes
- Interfaces with external systems

**Nonfunctional** Req’ts
- Accuracy
- Interoperability
- Modifiability
- Performance
- Portability
- Reliability
- Responsiveness
- Reusability
- Scalability
- Security
- Survivability
- Usability
… and then there are

- Vacations
- Holidays
- Sick Days
- Training
- Weekends
- Company meetings
- Department meetings
- Setting up new workstations
- Installing new versions of tools on workstations
- Troubleshooting hardware and software problems
But still … there is *Unfounded Optimism*

- The “Fantasy Factor” in the management ranks
  Managers want is *faster and cheaper*

- **Forms of optimism:**
  We’ll be more productive next time…
  A lot went wrong on this project, but next time…
  We started slow and were climbing a steep learning curve…
  we’ve learned so next time…

- **Collusion of Optimism**
  Developers are optimistic
  Executives like optimistic estimates
  For managers, optimistic estimates will be supported by upper management
Variation when numerous adjustment factors are present

100 groups of estimators using 17 effort multipliers

Estimating Session (number of groups in estimating session)
Low variation with a small number of adjustment factors

Only one!!

Estimate (staff months)

Estimating Session (number of groups in estimating session)
Average error from Off-the-cuff (WAG) estimates vs. reviewed estimates

Spend the time…
Unwarranted (silly) Precision

• More precise, but less accurate!

“Match the number of significant digits in your estimate (its precision) to your estimate’s accuracy.”
And don’t forget…

Other sources of error:

• Unfamiliar business
• Unfamiliar technology
• Incorrect conversion form estimated time to project time (e.g. assuming the project team will work 8 hours a day 5 days a week on scheduled and estimated tasks)
• Misunderstanding of statistical concepts
• Budgeting processes that undermine effective estimation
• Having accurate size estimates, but introducing error when converting the size to effort
• Overstate savings from new tools or methods
• Simplification of the estimate as reported “up”… that gets used for budgeting