Getting Better at Estimation

Carl Erickson
30 people, almost all developers and designers
10 years old, Extreme Programming practices from beginning
web, mobile, desktop and embedded
Fortune 500 to startups

What I’ve learned about estimation from being lousy to getting better

I’m going to start with a story...
Not so funny

I haven't looked at the problem. Completed: 0%,
Estimate: ~ 2 weeks

I've looked at the problem. Completed: 50%, Estimate:
~ 2 more weeks

I've implemented almost everything. All that remains is
the difficult stuff that I don't know how to do.
Completed: 90%, Estimate: ~ 2 more weeks

I've done everything. All that remains is doc, code
review, tests and error handling. Completed: 99%,
Estimate: ~ 2 more weeks

Adapted from James Iry's blog

This is funny, you'll probably laugh.

Imagine a project manager or customer asking you the questions,
“How far along are you? When will you be done?”

2 weeks later,... <click through>

If this represents your typical experience, it really isn't funny.
I guessed about half of you are lousy estimators

How many people think they’re good at estimating?

Can we ignore this?...
Why it matters

- Expectations are set by estimates
- Your professional reputation as a craftsman is on the line
- You might bind yourself contractually
- Business people rely on estimates to compute ROI
- Other business activities are planned on it

hard & fuzzy
rubs our nose in ambiguity, incomplete information
schools don't teach it
no one (I know) enjoys it

important things depend on our estimates

the good news is that you can get better...
You can get better

1. Using prior projects
2. Decomposition
3. Team estimation
4. Range estimates
5. Relative estimates
6. Measurement

I’ll talk about 6 ways of improving your estimates
I. Using prior projects

- From your time tracking / project management tool (right?)
- At a “bigger than a breadbox” level
- At a detailed story level

If you’re not tracking time, you’re not going to be able to compare your estimates to actuals and get better

You’re probably also taking a pretty big risk with your business

To demonstrate I pulled some recent projects and their hours from our time tracking tool...
Mobile Apps

Customer X, Mobile Member Card - 490 hours

Customer Y, myHealth - 800 hours

Customer Z, auto show - 450 hours

Customer A, Neocon iPad - 210 hours

Customer B, Bone Saw - 150 hours

Customer C, Event app - 370 hours

range of 150-250 hours
(bigger ones involved multiple platforms)
and some web apps...
Web Apps

Customer X - 1250 hours
Customer Y - 7200 hours
Customer Z - 4100 hours
Customer A - 30 hours
Customer B - 380 hours
Customer C - 3500 hours

big range
from a simple Drupal site to
complex functionality, with a million users

notice how you can’t use prior projects if you don’t track your time

suppose you’re going to invest more effort...
2. Decomposition

- To break something down
- Bigger means worse

The bigger the thing you’re trying to estimate, the more inaccurate the estimate will likely be

I have two examples, the first is real, the second, made up

Ready to do some estimation?

Please estimate the following...
AmazingApp.com would love to have an application made for the iPhone. Just a way for people to navigate easier on phone and watch movies. Is there any way you can provide us with an estimate? Thanks.

A. Customer

Sent from my iPhone
Write a program that sums the integers between 0 and 9.

It’s clear which you’ll be more accurate in estimating, right?
Exercise

• Decompose the task of making toast

Write them briefly on a card.
Select someone to read the tasks.
How many steps did you have?
Seems like a simple thing, right? Were you surprised at all about the tasks you found?

Another major gain from decomposition is the understanding gained through the task, particularly when you do it with another person.
3. Team estimates

- The people that do the work should make the estimates
- Estimates improve through independent thought and subsequent conversation

Why: we all know different things, have different experiences, think of different aspects

Exercise: Please read and follow the instructions on the next screen. I’m going to time how long it takes everyone. Ready?...
Clap slowly
in unison, with rhythm

N seconds

Humans want to converge, collaborate, work together

Lesson: not much really, for estimation, but I wanted to try this since I saw Seth Godin do it at BoSC.

Seriously: convergence is a problem in estimation. You want diversity and independence.

The problem: some people are more likely to speak than others, some have more “weight” than others.

Solution: planning poker...
Planning poker

1. The team discusses a task
2. Everyone, independently and secretly, estimates the task
3. Show your estimates, compare, discuss
4. Decide on the team estimate

Very simple: 4 steps

Exercise:
  Get together in groups of 3-4. (You don’t have to introduce yourselves)
  Estimate in units of hours.
  We’re going to do one discussion, and it’s going to be short.
    The task is so simple we don’t need to discuss it much (artificial)
    The task is to paint this wall.
  Everyone do your secret estimate now.
  Show, compare and discuss.

Poll some groups: how long? how much variation?
What determined the differences within the team?
tools? assumptions about requirements?
4. Range estimates

- A single point estimate is... irresponsible
- Probability distribution of software tasks
- Low and high
- Simple idea
  - aggressive but possible
  - highly probable

Shit happens. Stuff varies. Subtleties arise. You can’t foresee everything.

Tries to represent the natural spread of the work required.

A probability distribution function is how you model this
  - a graph of the probability of the actual value over the possibilities
  - <do a drawing>

Two estimates are better than one. The range recognizes the uncertainty.

So what should “low” and “high” mean?
  - depends on the shape of the PDF

Simple idea:
  low - aggressive, but possible
  high - highly probable

Think about: the iPhone app vs the Sum of integers app
  hugely different PDFs

Simple example: making toast...
The thing about software tasks is that the PDF is a “long right tail” distribution
Exercise: Making toast

1. select bread
2. insert bread, activate toaster
3. toast bread
4. retrieve bread from toaster
5. butter bread

we're going to use my project decomposition

aggressive but possible, highly probable

1. 5 seconds to decide which bread you want
   -unless, the one you select, open and select a slice is moldy
   -I've never had more than 2 moldy breads in my basket
   -so worst case is 3x or 15 seconds
2. 3 seconds to drop the slice, push the lever
   -no variation here
3. 120 second toast cycle
   -unless, the bread is particularly moist & robust, or the toaster
     is in a bad mood
   -sometimes I have to run the toast cycle again, so 240 seconds
4. 3 seconds to retrieve bread
   -unless, it's a short little bread that doesn't pop all the way out
     in which case, I need to find the wooden toast extraction tool
     -which I don't use that often, so it might be hiding, or dirty
     -worst case I can foresee, 60 seconds
5. 20 seconds to butter the toast
   -unless, the butter container is empty and I have to get
     butter from the refrigerator, or maybe even the freezer
     -that requires a trip across the kitchen, and a softening cycle
       through the microwave, so additional 40 seconds
     -total of 60 seconds

Now what?...
Two estimates - now what?

- $\sum$ aggressive but possible
- $\sum$ highly probable
- Responsibly Buffered Estimate
  - create a single project buffer
  - each task contributes
- making_toast_1.xls

What do you do with the range?
  - sum the “aggressive but possible” - trouble
  - sum the “high probable” - hysterical sandbagging

<open spreadsheet>

What if you need a single number?

You need to buffer your estimate...
Responsibly Buffered Estimate

- Neither a loser's game, nor hysterical
- Each task contributes something to buffer
- Project buffer gives you a single number
- making_toast_2.xls

Responsible == neither too low, nor too high; based on risk

The contribution of each task to the buffer is proportional to its uncertainty

Open making_toast_2.xls

Explain how buffer term is calculated (sum of squares)

So a responsible estimate for making toast is...
Making toast

- $\sum$ aggressive but possible = 151 seconds
- $\sum$ highly probably = 378 seconds
- RBE = 290 seconds

RBE lies in between sum of low and sum of high

Note: does not protect you from things you don’t know (scope change) or out-of-project risks (power goes out)

The next thing to consider in improving your estimates is the units to use...
5. Relative estimates

- Absolute: “6 hours”, “3 days”, “1 week”
- Relative: “twice as big”, “half as big”
- Exercise: How much do these boxes weigh?

Absolute estimates are...

Relative estimates are...

*It’s easier to make accurate relative estimates than absolute estimates*

**Exercise:**
- Two boxes. No information on contents.
- How much do they weigh?

**First: estimate in absolute terms (ounces, pounds, kilograms, etc)**
- Anyone?
  - assumptions needed, just like in software
  - incomplete information, just like in software

**Now: in relative terms (2x, 4x, etc)**
- Anyone?

**Conclusion:** relative is easier to be better than absolute

**But is that enough to improve our estimates?**
- no, since we don’t have an estimate in money-land

there’s also a tricky thing when it comes to estimating time...
Units of time

- “I worked 8 hours on this 5 hour task”
- “I got 16 hours of work done this week”

If you’re estimating duration of something then you’re using units of time.

Once you’re done with the task, you know how much time it actually took.

Both the estimate and the actual are in the same units (hours, days, etc).

This leads to confusion.

Using a different unit for the estimate solves the problem...
Pick a name, any name. Use these units for task estimates

So creating and styling a login page is a “2 point story” now instead of a “4 hour story”

Combine the ideas of relative estimates and arbitrary units

Progress is measured in points per unit time (day, iteration, etc)

A 4 point story is 2x as hard as a 2 point story
and 1/2 as hard as an 8 point story

Advice: simplify things by using a discrete set of estimates (powers of 2, Fibonacci series, etc)

One problem left to solve. No one has ever accepted a project estimate from me in points. Your boss probably won’t be satisfied knowing there are 1200 points left in your project.

We need to map from “developer land” (points) into “money land”...
6. Measurement

• “Money land” means time.
• Measure development velocity
• Completion date and cost from total points and velocity

How do we relate points to time (money)?

Easy: measure how much work we can get done over some unit of time.

These measurements are the “ground truth” we use to calibrate and make meaningful our accomplishments in relative complexity points

The rate at which we can finish tasks of a given number of points is our velocity

We measure points per iteration - the team’s accomplishment for one week

If all the work we need to get done in the project is consistently estimated, we can use velocity to predict time-to-completion and total-cost

Here’s a great thing: the absolute accuracy of our estimates doesn’t matter...
Two teams

- Team One estimates: 2, 4, 8, 16, 32, 64
- Team Two estimates: 10, 20, 30, ..., 100

The first team will probably have a velocity in the range of 15–25

The second team might have a velocity of 100

The first team might have 200 points in their backlog

The second team probably has a backlog of 1000

Does it matter? obviously not, they’re both going to get the same work done
Consistency is all that matters

All this hangs together like magic...
We’re more accurate making relative estimates

Velocity maps from dev–land to money–land

Measurement keeps us honestly aware of how things are going

The last idea I want to share with you isn’t really about estimating...
If we’re measuring our progress, we can report on it

A burndown chart is a simple way to report to your customers about the vital facts of your project:
- how much work you’ve done
- when you’ll be finished
- how much it’ll cost
- how scope has changed over time
- what the impact of team or scope change is

Some sample burndowns...