What Can Goldilocks Teach about Software Estimating?

You may not be aware that in 1837 when Robert Southey published his popular retelling of the Three Bears story, the U.S. experienced the “Panic of 1837,” a financial crisis that touched off a decade long recession featuring unemployment, pessimism, lowered profits/prices/wages, and blamed on domestic and foreign origins. While we might consider 1837 a simpler time - it was without modern conveniences like indoor plumbing, the internet, and supersonic travel – some aspects of human behavior and communication aren’t that much different today. I thought about this when I was keynoting the 20th anniversary EuroSPI2 conference (software process improvement) in Ireland, the same week that I read the following in the British press:

“The Department for Work and Pensions has dropped a coalition government scheme to avert software disasters from its £2bn Universal Credit programme” forecasting the cancelation of the largest ever agile software development project – a project now four plus years behind schedule with potentially billions of taxpayer funds at risk.

Software estimating can be a slippery slope – if it’s too optimistic, projects end up late and over budget, if it’s too pessimistic, projects aren’t funded, and if it’s just right – more often than not, we chalk it up to “just being lucky” after the fact. After decades of completed software projects, one would think that the process of estimating project cost, effort and duration should be common sense, yet routinely we get jolted by headlines showing the opposite. Why does this still happen? Maybe we can gain some insight from Goldilocks.

As you may recall from childhood, the story of Goldilocks and The Three Bears is about a young girl who is walking in the woods and stumbles upon a house and enters to find no one home. Feeling hungry she sees three bowls of porridge in the kitchen and takes a taste of each finding the first one too hot, the second too cold, and the third one just right so she eats it all up. The story goes that she finds three chairs and again, the first two are too big and the last one “just right” and she sits for a while; then wants to sleep and finds three beds, the first too hard, the second too soft, the third just right and she falls asleep only to be awakened by the return of the three bears who discover the intruder and chase her off. An abridged version, but you get the jist.

In my experience, software estimating is similar. As project managers and software professionals we strive to create realistic and reliable software estimates, but based on our experiences (punished for going over budget or under budget), we often end up polarizing our estimates. It’s similar to the Three Bears:

1. Overly Optimistic (Best Case) Estimate
2. Overly Pessimistic (Worst Case) Estimate
3. Just Right (Historical Based) Estimate

Let’s explore how these work in practice.

Overly Optimistic (Best Case) Estimate
I’m as guilty of this as anyone – I routinely think that tasks will take less time than they really do (even writing this blog post ended up taking more drafts and a lot more time than I anticipated!) Often the overly optimistic estimate works this way:

- **Bottom up detailed estimate based on tasks** – this is where we try and break down the project into a series of hundreds of subtasks that we can estimate using “experts” who have done the tasks in the past. Unfortunately, with bottom up estimates, we sometimes don’t scale our estimates based on the project size (no quantification of size) or use historical trend lines but rather the question “How much time do you think it should take to do this task?” We add up all of the optimistic – if everything goes right – independent tasks and can end up with an overly optimistic overall estimate.

- **Bottom up estimate cut in half** – this is where we’ve got through the process above and management (or someone with more influence than us) pushes us even further to reduce the cost and effort with a sweeping cut. Even a comment like “Okay this looks good, but we have to shave off another 30% from the budget,” can be commonplace.

With “rework” accounting for anywhere from 40%-60% of a project’s effort, it is folly to simply start from scratch and build such a bottom up estimate and expect it to be realistic. (Have you ever seen a bottom up estimate that accounts for rework?)

**Overly Pessimistic (Worst Case) Estimate**

When project managers or estimators “get burned” by an overly optimistic estimate gone real (real life events happening on the project), we often will regroup and swing too far in the other direction on the next project. In this situation, whether we do a top down product oriented estimate or a bottom up detailed estimate, we “pad” or inflate the components to account for “what if’s” that may not be realistic. It’s almost like we’re planning for an apocalypse project where the worst of the worst is going to happen – the result is an unrealistic budget and schedule that management often rejects. Even if we cite that “remember that xxx project where everything went wrong” – this is not the way to instill confidence in the project or get it funded. While it might seem prudent to load up the estimate with every possible risk (and embed costs for each) – it is highly unlikely that all of the risks will come true on a single project.

Often these estimates are not size or scope related but prepared based on previous project “disasters.” There is a better way.

**Just Right (Historical Based) Estimate**

In corporate America we like to talk about “lessons learned” but all too often we neglect such lessons when we estimate a new project (see 1 and 2 above.) “Just right” estimates are possible when we break down a project by work product based deliverables (similar to breaking down a construction project into roofing, plumbing, electrical components, etc.), quantifying the size of the product delivery, comparing it to completed historically similar projects and using a robust estimating model that takes into account projects in real life. For example, if we size a project at 1000 FP (similar to sizing a construction project of 1000 square feet), specify project goals (non-functional quality requirements for example), use a mathematical model based on actual completed project efforts, we are embedding real life behavior into our estimates. For example, if a given component of a particular size historically takes 200 hours to do the requirements through to testing (based on 20 similar projects), then it is likely that it will again take 200 hours on this project. To theoretically say it should take 100 hours (which it might if everything went exactly according to the project plan, which it never does) would ignore the lessons learned from the last 20 projects. This is like saying that Florida will never get hit by a hurricane this year despite 20 years of history that says this is highly unlikely.
SLIM provides such “just right estimating” capability in the palm of an estimator’s hands. Through thousands of completed real life projects and a robust parametric set of estimating scenarios, SLIM allows you to estimate and then perform what if analysis that tempers and augments your other (traditional 1. and 2. above) estimating methods.

Why keep scaring the Goldilocks in your corporation (the customers and executives) with estimates that just plain don’t work out? It’s about time we used lessons learned and come up with Just Right Estimates that mirror reality.