



Four Ways to Improve Project Performance by Avoiding Single-Point Estimates

White Paper

Abstract

The popular method of estimating with a single number (single-point estimation) has several interrelated and detrimental effects on project outcome. Estimate padding, estimate negotiation, and delayed resolution of uncertainty are all made worse by the use of single-point estimates. We examine the chain of cause and effect in each of these behaviors and suggest that estimating in ranges can produce significantly better project outcomes.

Discussion

What is an Estimate?

A dictionary definition of estimate: 1: the act of appraising or valuing. 2: an opinion or judgment of the nature, character, or quality of a person or thing <had a high *estimate* of his abilities>. 3a: a rough or approximate calculation. 3b: a numerical value obtained from a statistical sample and assigned to a population parameter. 4: a statement of the cost of work to be done. (Merriam-Webster Online Dictionary, 2008)

Throughout this paper we compare two different types of estimates: single-point and range estimates. A single-point estimate is an estimate like two days, or one week. A range estimate is an estimate like 1-3 days or 1-2 weeks.

It is a popular misconception that estimating in a range somehow lacks accountability or is in some way a mandate to “get it done whenever it gets done.” This misconception is perpetuated by a set of outdated tools that lack the ability to calculate schedules from ranged estimates. Lacking this ability, these tools force the user to fall back on single-point estimates. From this failing all manner of project ills spring forth.

At its heart, an estimate is just a statement of probability. An estimate is “good” if the actual result falls within the range of estimated outcomes. But the purpose of an estimate is not prediction.

The primary purpose of [...] estimation is not to predict a project's outcome; it is to determine whether a project's targets are realistic enough to allow the project to be controlled to meet them. (McConnell, 2006)

That is, good estimates do not need to be perfect predictors.

Estimating using Ranges

While informally polling people about their drive to work, the author has often heard ranges like “twenty minutes to an hour if the traffic is bad.” That is a factor of **three difference** between the best case and worst case estimates.

This is for a commute that the estimator knows precisely. A commute he has likely driven hundreds of times.

Should we expect the range of best case and worst case to be any less for a task which he has never performed?

No, we should look with suspicion upon an estimate with any smaller range.

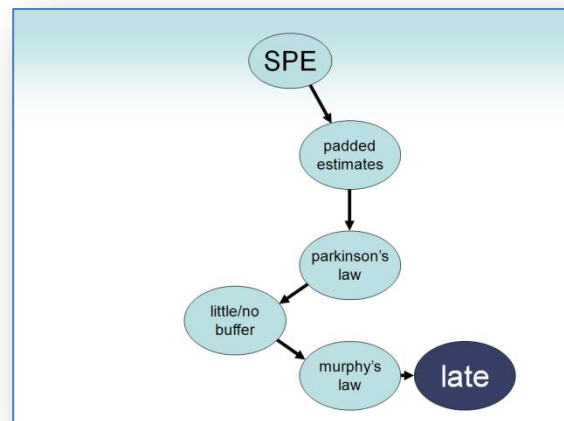
Rather, they need to give the project manager enough visibility and predictability to control the project and meet business objectives. As we will show, one critical aspect of visibility is the ability to see how much uncertainty is encompassed in a given estimate. This is impossible using single-point estimates.

Single-Point Estimates and Padding

When a person gives a single-point estimate (SPE) the estimator is saying, "I think it will take less than X amount of time to complete this task."

In this sense, single-point estimates act like promises. The estimator is making a social contract to get the work done in less than the estimated time. Since people are much more cautious about what they promise, they typically put in a task buffer. Putting it another way, the estimator attempts to pad their estimates. Yet projects are continually late. How can this be if there is all this excess padding?

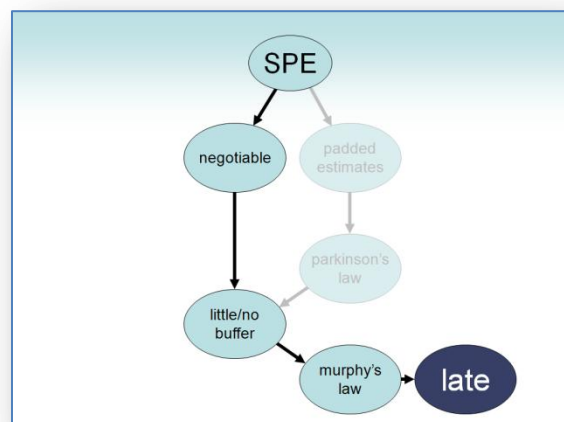
When it is known that there is the safety net of padding, it is tempting to either expand the scope of the work (Parkinson's Law) or put the work off until the last minute (Goldratt's Student Syndrome). (Goldratt, 1997) Both of these lead to having little or no risk buffer remaining when something goes wrong (Figure 1). Remember, that entire buffer was put in place to make sure that if something went unexpectedly wrong, the estimator would be able to recover. But now with little or no buffer left, Murphy strikes and he ends up late.



When the estimate is a range we naturally feel less need to pad the estimate. It is clear that we are not promising to absolutely get the task done in the shortest possible time. Instead we are giving our best effort to get the task done sometime within the range of the estimate. If we consistently come in near the top of our range, then we need to increase the size of our estimates in general. If we come in near the bottom of the range then we can safely decrease the overall size of our estimates.

Single-Point Estimates and Estimate Negotiation

The second problem with single-point estimates is that they are almost undoubtedly wrong. One can tell that they're wrong because if someone says something will take two days it is very unlikely that it will take *exactly* two days. The task in question may a little less time or it may take a lot more time. But we



can be nearly certain that it will not take *exactly* two days.

Since a single-point estimate is almost certainly wrong (and likely padded) it is open to negotiation. Negotiation seldom results in an estimate being increased. Thus these single-point estimates get negotiated down until it cannot be *proven* that it is impossible to complete the task in the allotted time. This is sometimes referred to as “the least defensible estimate”. (McConnell, 2006) This negotiation often ends up in the reduction or elimination of the risk buffer and with little or no buffer, Murphy strikes, and we end up late (Figure 2).

Giving the estimate as a range changes the dynamic of the negotiation. While the best case remains easily negotiable, the worst case is hard to argue (provided of course that it is reasonable and well thought out). Additionally, by using a range for the estimate in all documentation, the distinction is made clear between the estimate and the goal or target. Separating these two things can lead to earlier identification of projects that are unlikely to meet business objectives. After all, if the project is unlikely to meet the business goals then perhaps changes in scope, additional resources, or time on the schedule can be made part of the project plan.

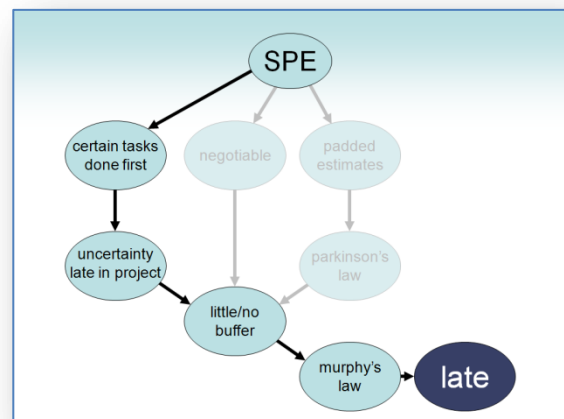
Beyond this, single-point estimates have another detrimental effect. Single-point estimates tend to obscure uncertainty.

Single-Point Estimates and Hidden Uncertainty

When given an assignment composed of a mixture of well-defined, cut & dried tasks and ill-defined, nebulous tasks, it is likely that the well-defined ones will be worked on first, given our natural tendency to want to “get something done.” However, this seemingly reasonable desire to get to work and start knocking tasks off your project list can unexpectedly lead directly to schedule slips. The reason is that the selection of well-defined tasks to begin working on causes a shortfall in remaining buffer near the end of the project. This can be particularly hard to detect if there is no way to measure and report the existence of this uncertainty (risk) in the project schedule.

All of the uncertainty in those deferred tasks is delayed until late in the project schedule. By that time, the remaining contingency buffer is too small for all of the retained uncertainty. But because uncertainty is so hard to detect when using single-point estimates, the project manager is often completely unaware that the schedule is very likely to slip. We cannot adjust the schedule to retain sufficient buffer and so end up with little or no buffer, Murphy strikes, and the project is late.

Had we used ranged estimates, the hidden



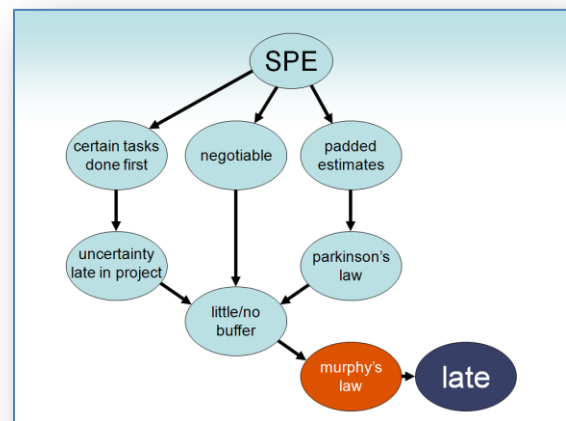
uncertainty would have been made manifest.

There is a notable difference between two tasks that have been estimated to take on average 4 days when one estimate is 3-5 days and the other is 1-7 days. The ability to see this difference (this increased uncertainty) gives us another dimension of information to help drive our decisions. We can begin to look at lingering uncertainty in a schedule or plan as what it is; added risk. Once we can observe the uncertainty we can begin to control it as well. Tasks with large amounts of uncertainty can be pulled forward in the schedule so that if something goes wrong in them there is still time (and buffer) left in the plan to recover.

Murphy's Role

Notice that all of these scenarios take us through Murphy's Law. Each starts with single-point estimates and, through various paths, ends up being late. All of them rely on something unexpected going wrong. But even if nothing goes wrong in an *individual task*, single-point estimates play another role in late projects, when multiple streams of work are going on in parallel.

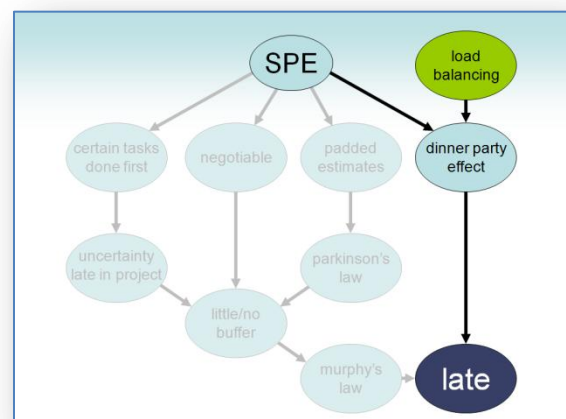
Those parallel work streams lead to load balancing -- which is good in and of itself, but which plays poorly with single-point estimates;



Load Balancing and Single-Point Estimates

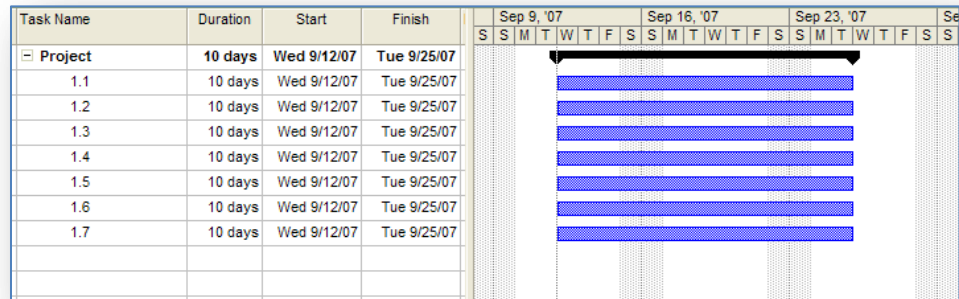
On any project it is a given that your resources (be they human or otherwise) are valuable. It makes sense to load balance where possible to prevent any resource from being idle. This is a good thing. But coupled with single-point estimates, load balancing leads us to what I call "The Dinner Party Effect."

Load balancing of resources on parallel work streams is a prerequisite to see this effect in action. Again, generally load balancing is a good thing. When a deadline is looming and one person is overloaded and likely to come in late, the best thing we can do is redistribute some of the overload to others on the team. This results (if you're a good and diligent manager) to all work streams being well balanced and finishing up at nearly the same time.



But the project is then highly likely to be late because of the interaction with single-point estimates. To illustrate, let's see how a classic project management tool schedules a completely load balanced project plan.

Here is a sample project plan consisting of 70 days of effort spread evenly across 7 people. Because we want it to be perfectly load balanced we give each person 10 days of work.



This schedule looks sane on the surface, but we would like to point out a couple of things at this point:

- 1) The schedule shows that each individual will finish in 10 days.
- 2) The project will finish in 10 days (because each person will be finished).

In order to understand why this is unlikely in reality, we will look at a simpler example involving the planning of a dinner party (and hence the name "dinner party effect").

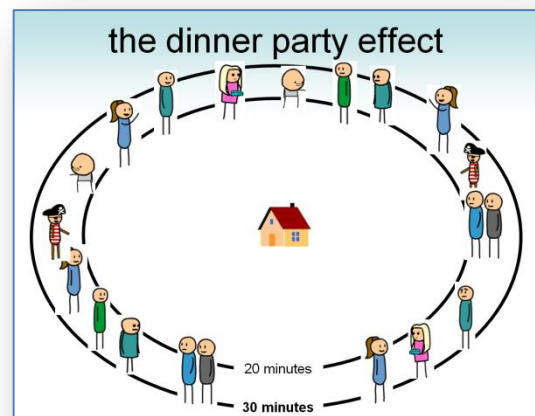
The Dinner Party Effect

Let us suppose that we invite 20 people to dinner at our house and we wish to tell our guests when to leave their houses to arrive on time for the soufflé to come out of the oven.

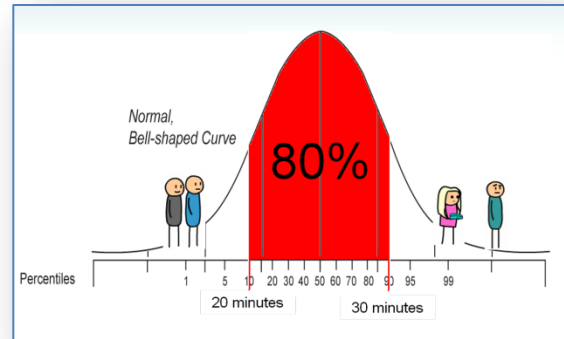
We happen to have a special set of friends who all live about the same distance from our house. Each of them estimates (with 80% confidence) that it will take 20-30 minutes to get to the party.

If we use single-point estimates we should be conservative and use the upper bound on the estimate to feed into a classic project management tool. The upper bound is 30 minutes, so if everyone leaves for the party at 7pm we can start dinner at 7:30. This is exactly the same as in the 70 day project we mentioned earlier.

When we look at this using ranged estimates rather than the artificial certainty of single-point estimates we see a very different picture.

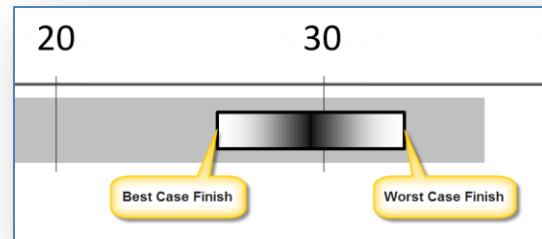


Using ranged estimates let us suppose that 20-30 minutes travel time is an 80% confidence estimate. That is to say, 80% of the time the travel will be between 20 and 30 minutes. Of the 20 people we expect to see 20% or 4 of them have travel times outside of this estimate. In the best case they are accurate estimators and we see two of them show up early. But we are virtually guaranteed to have at least two of them show up late.

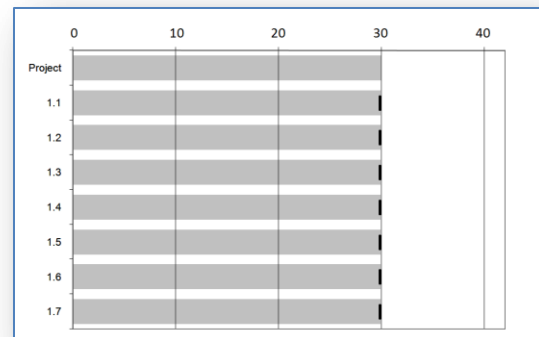


This is why dinner parties rarely start on time. It is not a problem of any one person being a bad estimator. Rather, it is an effect driven by the underlying probabilities of the system.

To demonstrate this we will employ a simple spreadsheet model using just seven of our dinner guests. This model has special bars on its Gantt chart which show both the best case and the worst case estimates for each item. In our case of the dinner party each guest is only 10% likely to arrive before the best case estimate, 80% likely to arrive *between* the best case and the worst case estimates, 90% likely to arrive before the worst case estimate, and only 10% likely to arrive after the worst case estimate. In the case of a single-point estimate the best and worst case estimates are the same.

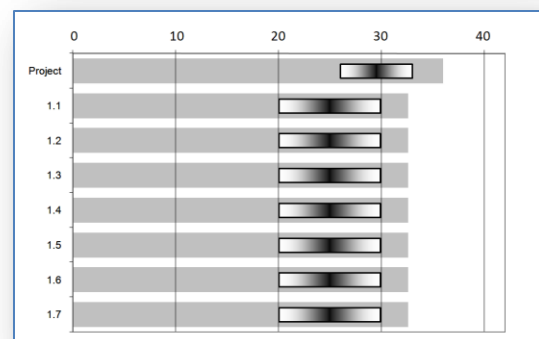


In the top plot we have essentially no uncertainty and everyone gets to the party in about 30 minutes. In the bottom plot the difference is that we have set the best case estimate to be *earlier* at 20 minutes. Common sense would say that this should move the project time *in* rather than pushing it later. *But it does just the opposite!!* Adding more parallel tasks makes this effect more pronounced.



This is a really exciting result!

It is exciting because even if we were managing projects well, our math was off so it's no wonder things ended up late. The tools we use can cause



project failure by forcing us to use single-point estimates.

Conclusion

Many tools make the use of single-point estimates inevitable. We have shown that single-point estimates can have multiple, deleterious effects on project schedules. Avoiding single-point estimates by simply giving your estimates in ranges can improve the accuracy of your schedules as well as the communication and interaction between project team members and project stakeholders.

While many of the effects of using single-point estimates assume at some level that something unexpected will come up in the course of the project, the dinner party effect is a purely mathematical (and therefore inescapable) outcome and in no way relies on Murphy's Law coming to make our lives miserable.

About LiquidPlanner

LiquidPlanner is online project management software designed to help teams of all sizes manage complex projects. It is an ideal project planning tool for teams who value efficient collaboration and project tracking. LiquidPlanner is built on an innovative scheduling engine that gives you a project schedule you can trust even before you know how long it will take to complete individual tasks. Robust yet easy-to-use analysis features provide a quick view of the project trajectory, from progress to resources to timelines, all in one place. LiquidPlanner also functions as scheduling software, with team management and resource scheduling tools that let you manage workflow of onsite or remote team members and adjust schedules as needs change. Global priority management gives you the freedom to set goals for your team or entire organization. Other collaboration features, such as wiki-like commenting and document sharing, keep vital project information flowing and safely stored in one location. LiquidPlanner is also flexible enough to handle all types of software projects, from scrum and agile methods to waterfall, as well as marketing projects, creative services projects, and professional services or consulting projects. LiquidPlanner is innovative, all-in-one project software.

About the Author

Bruce P. Henry | Director of Rocket Science

Bruce earned his title by completing an undergraduate degree in Math and both undergraduate and master's degrees in Physics. After sensibly deciding not to complete yet another degree, his doctorate, Bruce began his technology career as a Software Test Engineer for Microsoft. He later joined Expedia as a Software Test Lead, moving up through the ranks to serve as Director of Release



Technology and, ultimately, Senior Director of Quality Management before joining former colleagues Charles and Jason at LiquidPlanner. An experienced software development executive, Bruce has a keen interest in entrepreneurial ventures, startups, and building high-performing teams.

Works Cited

Goldratt, E. M. (1997). *Critical Chain*. Great Barrington, MA: North River Press.

McConnell, S. (2006). *Software Estimation: Demystifying the Black Art*. Redmond, WA: Microsoft Press.

Merriam-Webster Online Dictionary. (2008). *Merriam-Webster Online Dictionary*. Retrieved December 9, 2008, from estimate: <http://www.merriam-webster.com/dictionary/estimate%5B2%5D>