# Either your estimates suck or your job does

I hate when estimates come true consistently. As an engineer, I believe that if a job is reliably predictable, then it can also be easily automated. And as a project manager, I also believe that no one should spend time on a problem that a computer can solve for them. We all have roughly 80x365x24 = 700800 hours on this Earth, so making someone spend an hour of their life on some monkey work is a little more than a micro-murder.

Luckily, most estimates never come true, and I have a dozen polynomial models to show you why.

### The problem

A ditch digger digs a one-meter ditch in one hour. How many meters a ditch digger will dig in eight hours?

// plot

Eight, right? If you believe that actual people act like ditch diggers from a third-grade mathematical problem, read no further. Seriously, don’t bother. Just enjoy your C-level management position.

But you chose to read further after all. This means that you know a thing or two about people, mathematical modeling, or actual digging. Yes, small efforts don’t scale. Nobody can dig for eight hours straight without rest, people get tired. There is also heat and rain, there are tree roots, rocks, clay. The shovel can break. There could be a power cable underneath. This all makes work estimation hard and, at some point, impossible.

## Non-linearity of the predictive model

Obviously, ditch digging is not an entirely linear process. Intuitively, an eight-hour ditch should be somewhat shorter than eight one-hour ditches. But how shorter exactly? Let’s gather some data and build a better model.

So let’s say we know that a three-meter-long ditch takes four hours to finish. Good! This gives us a new data point and allows us to promote our model from linear to quadratic.

// plot

Hold on! But now, when we have tiredness accounted for, it looks like, after the seventh hour, a digger starts putting dirt back into the ditch for some reason. This doesn’t seem right. Perhaps, tiredness in itself is not a linear process either. You can’t get into negative efficiency just by getting tired.

Well, of course, this happens all the time in software engineering. Well-rested people write code, tired people write bugs. But in ditch digging, you don’t just undo your work when you’re tired, so looks like we need a better model still.

We can’t constrain a polynomial from going down, but we can dig one more ditch, add another data point, and make the polynomial model look more like a process we want to copy. Let’s say we made an experiment, worked for 3 hours straight, and dug ourselves a 2.5-meter ditch. Good!

// plot

So to build a convincing polynomial model of a ditch digger, we need at least three data points. This means that the estimating person should have dug at least three ditches in their life.

Intermediate conclusion: to build a predictive model, you need data.

### Multiple factors (input variables)

Ditch digging is impacted by multiple factors. Is the ground rocky or sandy? Are there tree roots? Are we digging with shovels, or do we have a digging machine? Introducing each new factor effectively introduces a new variable, a new dimension to the problem.

So to build a ditch digger’s model with one variable – time – you need three data points. That’s three ditches.

// plot

Now to account for two different types of soil, you need six data points. Three for rocks, and three for sand.

// plot

To add tree roots into the equation, you need, once again, to duplicate your data set, because roots can grow between rocks and in the sand too.

// plot

And, of course, having a digging machine is a game changer, so, once again, you have to duplicate your data set just to build a comprehensive model.

// plot

So the three points from before now become 24, which means that the estimating person should have dug at least 24 ditches before they could build a good enough model in their head. That’s a lot of digging.

Also, factor-wise, we barely scratched the surface. Some factors are not entirely accountable and not even measurable. They just contribute to the general unpredictability.

Intermediate conclusion: the more factors the model accounts for, the more data you need. Not just more but exponentially more.

### Input error

When we say, a meter takes an hour to dig, we don’t mean exactly 100 centimeters and exactly 3,600,000 milliseconds. It’s roughly an hour for roughly a meter. This roughness means that when we dig a ditch, the result fluctuates from time to time, and instead of a specific meter per time function, we have a range of plausible functions. Our model is not a curve but a bundle of them.

For a linear model, this is not that bad. We allow a 10% error on the data point, and with this error, the model starts to diverge with time but it still retains some predictive power.

// plot

But we already know that digging is non-linear so let’s try a two-point model, a quadratic one.

// plot

Well, this is worse. The same 10% error on input results in a +/- 2 meters at the end of the time scale. But wait! As we add points, it gets better!

// plot

Sorry, I meant “worse”. The model gets worse still. The effect of non-linear models being vulnerable to small input errors is pronounced much better though, and that’s what I meant to say. With a three-point or cubic model, a modest 10% input inaccuracy results in the model being completely useless.

We can mitigate this effect by narrowing down the input error ranges. We can do several measurements at each point, and then do some statistical analysis: weed out the outliers and compute the real confidence interval instead of some hypothetical 10%. We have math to do so. But! Once again, we need more data.

Intermediate conclusion: we need even more data given that the input for our model is inherently inaccurate.

### What’s my point?

There is a notion that estimation is a skill. To develop this skill, you need to build some kind of model in your head. You need data. And non-linearity, multiple factors, and flimsy measurement make this modelling difficult and, now this is my point: **in most and not just a few exceptional cases, impossible**.

You can say that by adding more data, we can make every model better. Sure, that’s true. That’s what we do in machine learning as well. But we need tons of data. And every data point – is a job done in the past. How many user stories can you close, how many bugs can you possibly fix in your lifetime? Remember, 700800 hours, and that’s it.

And that’s why your estimates suck. Just like everybody else’s. None of us have enough experience to develop a plausible model for generic software engineering work. To do that, we need way more data than we can possibly gather in our lifetime. Unless the work we’re trying to model is somehow linear and isolated from all the possible impacting factors. And if it is, it can and should be automated.