title: **How to give realistic estimates in front of the team?**
num_comments: 10
num_up_votes: 6
upvote_ratio: 0.88

#### Post Text ####

I would think I'm a seasoned developer, been in the sector on and off for more than 10 years. The way our team works is by giving points to the different tickets (kind like 1, 2, 3, 5, ...) in front of all the team, nothing new I would think. I tend to underestimate tickets, there is a self-guilt issue but also because I tent to now account for the unexpected, I understand that I'm not doing a favor to anyone by underestimating but not matter the experience I have I keep falling on the same issues. Now I will be starting on a new team next week, Does any of you has any process for estimating that has helped them give a more accurate view on what a ticket might take?

#### Comments ####

###### Commenter_2
ID: REDACTED! ~(o.o)~ <3, Upvotes: 7
If you know you typically underestimate you could simply multiply by a safety factor before giving your number.

Another approach is to look for aspects of the work that are unknown to you before estimating. Are there new technologies you haven't worked with before? Is there a part of the code you haven't touched recently? All these and other factors will increase the time taken.

You could keep a log of what your estimates were and how long the tasks actually took, and the reasons why if the tasks took longer. Reflecting with this log could help you improve your estimates over time.

> ###### Commenter_3
> ID: REDACTED! ~(o.o)~ <3, Upvotes: 4
> I second starting a log of your task estimates and real-world times. Only you can work out your personal talents, tarpits, and coding effort.
>
> Then as a general rule if you have to give an estimate: Make a ballpark from that log knowledge. Take that ballpark estimate and double it. Nobody is ever unhappy if you finish something early (or take some personal time) but over-promising can lose you a contract or job. If possible describe tasks in terms of days rather than hours.
>
> > ###### Commenter_4
> > ID: REDACTED! ~(o.o)~ <3, Upvotes: 1
> > oh I like this, my 'talents' and my 'tarpits'
> > it's both shocking and so so accurate... mmm yes

###### Commenter_5
ID: REDACTED! ~(o.o)~ <3, Upvotes: 5
As unhelpful as this may be, my experience with estimating sprint work is that the best estimate is no estimate. It's super difficult to get right and often things can't be accounted for or planned for until you start looking into the ticket

###### Commenter_6
ID: REDACTED! ~(o.o)~ <3, Upvotes: 5
I legit always double my estimates and usually it works out well.

###### Commenter_4
ID: REDACTED! ~(o.o)~ <3, Upvotes: 3
yep. I actually do a 2.5x on what I really think. It's shocking how often I burn up that 2.5...

###### Commenter_7
ID: REDACTED! ~(o.o)~ <3, Upvotes: 2
If you're using story points (0.5, 1, 2, 3, 5, 8, 13, 20, 100, etc) then the lead *should* be starting with a ticket that's somewhere around 3 points, and then you should be estimating difficulty relative to that ticket. It's much easier to say it's a bit or a lot more/less complex than something else, than it is to estimate individual tasks in terms of time.

If they're not, then just do it yourself. After one is estimated, think about the next ones in terms of how difficult they are relative to one another, and give them a value based on that.

###### Commenter_8
ID: REDACTED! ~(o.o)~ <3, Upvotes: 2
I subscribe to the Montgomery Scott Method of Task Time Estimation: Take the amount of time you think it'll take, and triple it. If something goes wrong, you've got a huge buffer. If you get it done fast, you'll be a hero.

Everything I needed to know I learned from Star Trek :)

###### Commenter_9
ID: REDACTED! ~(o.o)~ <3, Upvotes: 1
If I'm familiar with the area that something will be working in I will add difficulty based on that. For example, one project had a massive JS file that was basically just jumbled spaghetti code for one section of the website. If we ever more than touched that I automatically went up one Fibonacci difficulty (ie. 1 -> 2, 5 -> 8, etc.) because I knew that besides how hard it would be to do the base work, you also would need to slog through the swamp that was that file and it would be harder by virtue of that.

###### Commenter_10
ID: REDACTED! ~(o.o)~ <3, Upvotes: 1
Pointing is different for every team, so exact recommendations from us may not really work.

However! My team made a pointing guide for any new people that join. I recommend reaching out to your scrum master to have a session for this! It's really helpful. You just go through points 1-8 (or however high up you guys normally estimate in a sprint) and have everyone explain what type of work they think fits that point. They can even have examples for each one. Then you can reference it later and estimate accordingly.