[Story Estimation - sizing and splitting](http://being-sm.blogspot.com/2015/05/story-estimation-sizing-and-splitting.html)

I have found that a lot of people have problems with sizing / estimation meetings, so here are some thoughts on that topic. Let's start by defining Story Points - the unit of estimation. To put simply - *story points are a measure work effort.*We use units of work effort to derive time.

**Why estimate?**

In my opinion, the greatest need for estimation primarily arises from the need for release planning.  Most teams I have worked with work on some sort of a plan for releasing their software to the users. Estimates can provide data about how you are doing with reference to your plan.

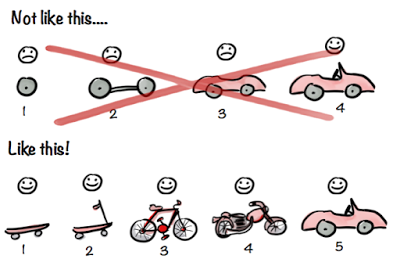
Not everyone may need estimates though. If you are experience in Agile, consider giving up estimates for a release or two. However, if you are starting out on Agile, I recommend starting on the side of having estimates.

**Sizing defects?**

In my opinion, if a defect must be fixed for the release to go out, it doesn’t matter if you estimate it or not. If you are want to gauge how many low / medium priority defects you can fix in a given time frame, it may make sense to put sizes on them.  
  
Defects can be notoriously hard to size, so time-box the sizing activity.   
  
Now that we've covered the basics, lets talk a bit about what the Scrum Master needs to do in order to have a successful sizing meeting.

**Coaching the product owner on Minimum Viable Product:**

I think this image is a perfect example of Minimum Viable Product. I'll let you think about what it is saying. All I can say is that I'm a big proponent of starting out by trying to build stakeboards, not spaceships. Coach your product owners to do the same.

[](http://2.bp.blogspot.com/-bpy2agcybFo/VVvPhNp-afI/AAAAAAAAEB8/5i-72hNZ9TU/s1600/sizing%2Band%2Bsplitting.png)

I came across this example from the wonderful podcast at - [This Agile Life](http://www.thisagilelife.com/58/)

**Coaching the team to embrace unclear requirements:**

At the outset of a project, I tell my team the following 2 points:

* Inspiration does not come in the form of a prioritized backlog.
* You don't really know the full potential of your product until you've built it.

An example of product potential is - I feel confident in saying that Twitter didn't start out trying to build a tool to overthrow dictators. Most products aren't used the way they are initially defined. This makes the product definition job a really hard one. When building a new product, there is no right or wrong answer. This is a wonderful reason to embrace Agile. It is the job of the team to shape unclear product requirements / stories and help build small increments of functionality. Sizing aids that process. Don't shy away from it, even though it can sometimes be painful.

**Mechanics of sizing and splitting meeting:**

**Reference Story(s):**

I recommend defining at least one, possibly 2-3 standard / reference user stories. User stories in the product backlog should be estimated based on these reference stories. A reference user story can be story that is fairly well understood by everyone in the team. A story that is already implemented by the team can be a good candidate reference story. It is best if it touches all the layers - UI, business logic, database / server.

**Some points that aid sizing / estimation meetings**

I've been a part of teams where team members cannot take their engineering hats off when putting a number on the story. For whatever reason, they are scared to put numbers on user stories. I call this "everything-is-an-5" syndrome. I use an Earth vs Sun example to try to overcome this. Even though the Earth is much smaller than the Sun, it doesn't mean there is no work involved in building Earth. It is just that it is less compared to building the Sun. So Earth and Sun can't both be sized to 5.

**Sizing  based on "what?" and not "how?"**

I encourage the team to size based on the “what?” and now the “how?” Let's take an example: Let’s say our reference story is one where - we top off the coolant in a car. The story we need to size is say - changing brake fluid.

I have seen teams get stuck in discussing the implementation details of the brake fluid user story. This has the potential to make sizing meeting remarkably painful. To overcome this, I explain to the team that the question "what?" should yield the size of the story, while the question "how?" is solved in the sprint itself.

To help aid sizing based on "what?", I encourage teams to try to take a step back and understand what we are changing / touching with the story being sized. Once that is clear, I compare and contrast the story being sized with the reference story.

For our brake fluid story story, I can say that the car will have some sort of a container holding the brake fluid, just like the coolant container in the reference story. The reference story only involves topping off the coolant container, while the brake fluid story is about changing the fluid. Based on this information, we can say that the brake fluid story is bigger than reference coolant story. This level of detail is generally enough to put sizes / estimates on stories.

I find that using this technique has yielded reasonably quick and usable estimates.

In addition to the points above, I found [this article](http://www.mountaingoatsoftware.com/blog/how-can-we-get-the-best-estimates-of-story-size) by Mike Cohn really helpful to have an "Estimation 101" workshop with my team.