Top 10 Agile Estimation Best Practices

1. **Use more than one person** – By engaging the team in the estimation process we gain the benefits of additional insights and consensus building. Additional people bring different perspectives to estimating and spot things individuals may miss. Also, the involvement in the process generates better consensus and commitment for the estimates being produced.

2. **Use more than one approach** – Just as one person is likely to miss perspectives of estimating so too are single approaches. Use multiple estimation approaches (comparison to similar projects, bottom up, user story points, etc) and look for convergence between multiple approaches to reinforce likely estimate ranges.

3. **Agree on what "It" and "Done" means** – make sure everyone is estimating in the same units (e.g. ideal days), have the same assumptions, and are based on standard developer ability/effort. When asking for estimates spell out what you are asking them to estimate. What does "Done" include? Coded, unit tested? How about integrated and system tested? What about refactoring contingencies? User meeting time?

4. **Know when to stop** – estimating an inherently unpredictable process (custom software development with evolving requirements) will never be an exact science. Balance enough effort against the diminishing returns and false accuracies of over analysis. Look for broad consensus between team members at a course grained level and then move on. It is better to save estimation time for periodic updates than over analyze.

5. **Present estimates as a range** – We call them "estimates" not "predictions" because they have a measure of uncertainty associated with them. Manage the expectation of project stakeholders and present them as a range of values. E.G. Between $90,000 and $120,000

6. **Defend / explain estimate range probabilities** – If stakeholders automatically latch onto the low end of an estimate range explain the low probability of achieving this and steer them to a more likely value. If you organization persistently fails to understand, present a range of likely values (e.g. around the 50% to 97% probability range)

7. **Don't reserve estimating for when you know least about the project** – Estimation should not be reserved for the beginning of projects. Instead done throughout as we learn more about the emerging true requirements and ability of the team to build and evaluate software.

8. **Be aware of common estimation omissions** – Consult lists of common estimating omissions (such as Capers Jones') and ensure these items are taken into account. Look back at retrospective notes for things that did not go so well, and tasks that were missed or ran late – make sure we include enough time for these.

9. **Embrace reality early –** As the project progresses**,** it is tempting to think development will get faster and faster now all the technical problems have been overcome. However don't under estimate the load of maintaining and refactoring a growing code based. Especially if the system is now live; support, maintenance, test harness updates, and refactoring can quickly erode the velocity improvements anticipated, so use the real velocity numbers.

10. **Review, Revisit, Remove head from sand, Repeat** – Our first estimates will likely be our worst. Don't leave it there; review the project velocities to see how fast we are really going. Revisit the estimates armed with the real velocities to determine likely end dates. Embrace the reality you see, "The map is not the territory", reprioritize and repeat the estimation process often to adapt and iterate to the most accurate estimates.