Three methods of budgeting and scheduling

1.A traditional approach

2.An agile approach

**3.A middle of the road approach**

* Comparing the st [How to play planning poker?](https://wingman-sw.com/papers/PlanningPoker-v1.1.pdf) (1 hour).
* Read:  [Agile estimation](https://www.deltamatrix.com/agile-estimation/) (1 hour).
* Read:  [Story Points vs Function Points](https://www.scopemaster.com/blog/story-points-compared-with-cfp/) (1 hour).
* Read:  [No estimate method](https://plan.io/blog/noestimates-6-software-experts-give-their-view/) (1 hour).
* Read:  [Comparing Approaches to Budgeting and Estimating Software Development Projects](http://www.ambysoft.com/essays/comparingEstimatingApproaches.html) (1 hour).
* Read:  [Critical Path](https://pmstudycircle.com/total-float-versus-free-float/) (1 hour).
* Read:  [Tips for Project Planning with Gantt Charts](https://www.smartdraw.com/gantt-chart/gantt-chart-tips.htm) (1 hour).
* Read: [The Ultimate Guide to the Critical Path Method](https://www.smartsheet.com/critical-path-method) (1 hour).

I think Planning Poker are easier and reasonable to evaluate methods for website development.

A middle of the road approach to requirements enables you to take an easy approach to budgeting and scheduling because of the higher level of control it provides business stakeholders:

1. Without much effort, Stakeholders can create reasonable estimates and schedules

2.Stakeholders in complete control over how much money is spent, for how long, and on what.

Premise:You have made enough requirements, envisioned the scope of the project, and done enough architectural modeling to determine the potential strategy for building the system.

Condition：

1.gather a small group of people who have relevant experience from similar projects in the past, and more importantly have a stake in the new project

2.Spend less on technical staff budget and early R &amp; D budget.

3.Enterprises want and need speed and predictability. Some #noestimates critics believe that software practitioners should learn better estimation skills to improve our predictability. Given that as software practitioners, we must make a lot of estimates, it seems a good idea to learn and use more effective estimation techniques.

4.Gather enough useful and consistent information about all these projects to meaningfully integrate into your decision-making.

5.The end result is that developing working software via a test-first approach and details captured in the form of executable specifications These narrowing estimates are referred.