

Team Estimation

Introduction

“What is the size of that feature? That story? That impediment? Is it a 2, a 3, a 20?” How do you tell? What does that number mean? It must be important because teams spend a lot of time trying to estimate their work. And yet, for all that work, the numbers aren't all that precise. Especially early in a team's life, they often have to go back and have to redefine what they mean. The problem is not that the team doesn't know what they are doing; rather, they are being asked to be precise beyond their knowledge and experience. They need an approach that cooperates with them where they are.

Team Estimation is a two-stage approach to estimation that lets teams build estimates while they are making sense of the work before them. It recognizes that people find it easier to compare the relative complexity of one feature or story with another even if they do not yet know all aspects of that feature or story. After sorting them into buckets of relative complexity, then the team can assign a number that reflects the “size” of the stories.

This approach helps teams avoid the usual traps of estimating: getting bogged down by too much detail, turning estimation into design, trying to get too precise, and that level of discomfort that makes it hard to commit. Instead, the team quickly gets the information they need to decide what to work on and how much to commit to in current the planning horizon.

Team Estimation is fast, easy and fun. Based on an idea by Steve Bockman, it is useful both for new teams who are still learning their capabilities and capacities and more experienced teams who more or less know the size of “a feature of this type.” In fact, Team Estimation can help those more experienced teams avoid mistakes by making too many assumptions.

Why Estimate?

Estimates represent a team's best guess about the effort required to complete a work item (feature, story, etc.) based on what they currently know. As they gain more experience, they will have more information and can then refine the estimates.

The best estimates are ones that take into account the complexity of the work item. More complex work items require more time, effort, attention, and testing. One good way to think complexity is the degree of connectedness of the feature or story. The more interconnected it is with itself or the more connection it has with other work items, the more complex it is.

In an Iterative Agile development approach, the more you are able to make work items the “right size” or close to what the flow of work will allow, the more likely you are to perform the work without overloading your capacity. If an item is too complex, too big, then if you can decompose it into less complex pieces, you will have an easier time of it. In a kanban development approach, the size of the work item helps determine the class of work and corresponding policy, service level agreement, and average cycle time .

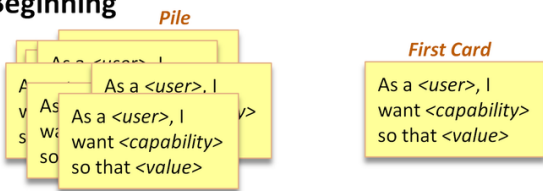
How to Play the Team Estimation Game

Traditionally, this game is played at the iteration planning session to size stories for the next iteration. However, it can be used by any team that needs to estimate work items. For example, a team of Product Champions have used this approach to prioritize and plan features across the portfolio of projects.

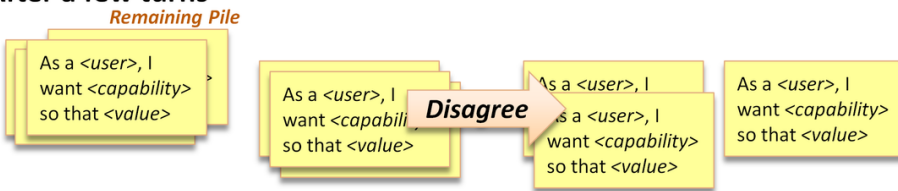
Step	Do this...
Setup	<p>The complete set of cards (for what you are estimating: features, stories, etc)</p> <p>The playing surface. Depending on the size of the group, this might be a table, a white board, or a computer screen if you have a dispersed team and have a suitable electronic tool.</p>

Step	Do this...
First play	The team facilitator or team member takes the top card off the pile and places it somewhere on the playing surface, about a foot away from the pile.
Relative estimation of the rest of the deck	<p>The team facilitator or a team member turns over the next card. As a team, they agree where to place the card relative to other cards on the playing surface:</p> <ul style="list-style-type: none"> • <i>Left</i> means it is relatively less complex • <i>Right</i> means it is relatively more complex • <i>Underneath</i> or <i>in the same pile</i> means it is of equal complexity <p>Rule: You may move an entire stack of cards left or right if you need to create a new stack whose complexity is in-between those other stacks</p> <p>Rule: Anyone may talk about why cards are being moved or about what they think about size of the work items. The goal is to get clarification and not to get too hung up on the exact sizes of the estimates. Remember that these are just estimates of relative complexity!</p>
Assign points	<p>After all cards have been placed, the team works together to assign points to each stack to indicate the size of work items that are in that stack. Use the sequence: 0, 1, 2, 3, 5, 8, 13, 20, 40, 100, 200, 400, and 800. When done, write the assigned points on each card. If there are multiple cards in a stack, each card gets the same number.</p> <p>Rule: The points represent complexity, not time.</p> <p>Rule: The team decides what the numbers mean to them. However, like a logarithmic scale, make sure each number is significantly larger than its predecessor. complexity than the previous one: “three” represents significantly more complexity than “two,” and so on.</p>

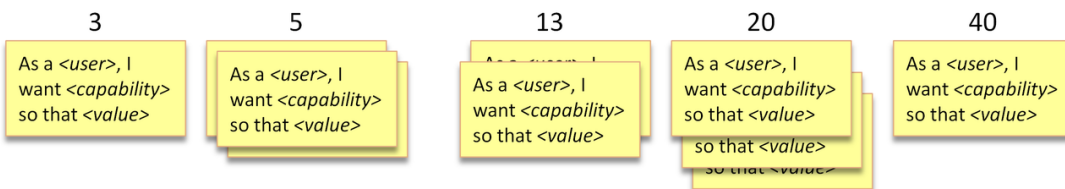
Beginning



After a few turns



At the end: Assign Points



Business-Driven Software Development (BDSD) is Net Objectives' proprietary integration of Lean-Thinking with Agile methods across the business, management and development teams to maximize the value delivered from a software development organization. BDSD has built a reputation and track record of delivering higher quality products faster and with lower cost than other methods

BDSD goes beyond the first generation of Agile methods such as Scrum and XP by viewing the entire value stream of development. Lean-Thinking enables product portfolio management, release planning and critical metrics to create a top-down vision while still promoting a bottom-up implementation.

BDSD integrates business, management and teams. Popular Agile methods, such as Scrum, tend to isolate teams from the business side and seem to have forgotten management's role altogether. These are critical aspects of all successful organizations. In BDSD:

- **Business** provides the vision and direction; properly selecting, sizing and prioritizing those products and enhancements that will maximize your investment
- **Teams** self-organize and do the work; consistently delivering value quickly while reducing the risk of developing what is not needed
- **Management** bridges the two; providing the right environment for successful development by creating an organizational structure that removes impediments to the production of value. This increases productivity, lowers cost and improves quality

Become a Lean-Agile Enterprise

All levels of your organization will experience impacts and require change management. We help prepare executive, mid-management and the front-line with the competencies required to successfully change the culture to a Lean-Agile enterprise.

Prioritization is only half the problem. Learn how to both prioritize and size your initiatives to enable your teams to implement them quickly.

Learn to come from business need not just system capability. There is a disconnect between the business side and development side in many organizations. Learn how BDSD can bridge this gap by providing the practices for managing the flow of work.

Why Net Objectives

While many organizations are having success with Agile methods, many more are not. Much of this is due to organizations either starting in the wrong place (e.g., the team when that is not the main problem) or using the wrong method (e.g., Scrum, just because it is popular). Net Objectives is experienced in all of the Agile team methods (Scrum, XP, Kanban, Scrumban) and integrates business, management and teams. This lets us help you select the right method for you.

<p>Assessments</p> <p>See where you are, where you want to go, and how to get there.</p> <p>Business and Management Training</p> <p>Lean Software Development Product Portfolio Management Enterprise Release Planning</p>	<p>Productive Lean-Agile Team Training</p> <p>Team training in Kanban, Scrum Technical Training in ATDD, TDD, Design Patterns</p> <p>Roles Training</p> <p>Lean-Agile Project Manager Product Owner</p>
--	---

