Agile Estimation: Key Principles and Practices for Successful Agile Projects

John Clifford
Senior Fellow
Construx Software
Copyright Notice

These presentation materials are © 2001-2012 Construx Software Builders, Inc.

All Rights Reserved. No part of the contents of this presentation may be reproduced or transmitted in any form or by any means without the written permission of Construx Software Builders, Inc.
Agenda

- What Is An Estimate?
- Why Estimate?
- When To Stop Estimating…
- Confusing Precision With Accuracy
- The Cone of Uncertainty
Agenda (continued)

- Estimates, Targets, and Commitments
- Agile Estimation Basics
- Story Points: What, Why, and How
- Plan Using Estimates, Commit Based Upon Demonstrated Performance
What Is An Estimate?

Es · ti · mate:

*noun*: An approximate calculation or judgment of the value, number, quantity, or extent of something.

“An estimate is the most optimistic prediction that has a non-zero probability of coming true.” – Tom DeMarco

Oxymoron: *Accurate Estimate!*
Why Do We Estimate?

- Predictability is the primary goal of project management
  - What will it cost?
  - How long will it take?
  - What will be delivered?

- We estimate to determine what we do next

- Most organizations spend too much time and effort estimating ineffectively

*In the desire for predictability, organizations often value precisely wrong estimates over approximately right ones*
The Effort-Return Curve

- Most of the value comes from the initial part of the effort.
- After a certain point, additional effort produces no additional value.
- Stop after the ratio drops below 1 (Pareto Principle).
All estimates have uncertainty, whether implicit or explicit

- **Precise** estimates are represented with a fine level of granularity
- **Accurate** estimates have a range that includes the actual value

*It’s better to be accurate than precise, so don’t confuse the two*
Precise or Accurate?

- Population of New York State (USA) in 2010
  - Between 10 million and 30 million people
  - Two to three times the population of New York City
  - 20 million people, plus or minus a million
  - 19,378,102 people
The Right Tool For The Job…

The Level of Precision of your estimates should correspond with the Level of Uncertainty

- Project-level estimates
  - Team-sprints, team-months, team-quarters
- Feature-level estimates
  - T-shirt sizing (buckets)
- Backlog Items
  - Story points
- Item component tasks
  - Ideal remaining hours
The Cone of Uncertainty

Source: *Software Cost Estimation with Cocomo II* (Boehm 2000)
Confusing *Estimates* with *Commitments* is commonplace, and leads to failed projects

- **Estimates** are the responsibility of the people doing the work
- **Targets** are set by the people paying for the work
- **Commitments** are agreements between the people doing the work, and the people paying for the work
  - Commitment is achieved through negotiation
Agile Estimation Basics

“Estimate effort, derive duration”

- We estimate effort for each Backlog Item in terms of *Story Points*

- We estimate *Velocity*, the rate at which we can implement functionality, in terms of *Story Points per Sprint*

- We derive *Duration*, how long the project will take, in terms of *Sprints to Completion X Sprint Duration in Weeks*
Story Points: What Are They?

- A measure of relative size/effort/complexity
- *Not* a measurement of duration!
- We use arbitrary numbers to reflect the imprecise nature of estimation
  - Fibonacci series
    - 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, ...

“Estimate effort, derive duration!”
The ‘Why’ of Story Points

- Effort is a constant; duration is dependent upon who does the work, and how they do it

- Teams can improve the rate at which items can be implemented; they can’t work more than 24 hours in a day

- Fibonacci Numbers imply a range (0.6x to 1.6x), allowing us to take advantage of the Law of Large Numbers

- The ‘fuzziness’ of story points keeps us from wasting time trying to unnecessarily refine estimates
Story Point Estimation How-To…

- Pick one Reference Story (Backlog Item) that the team understand and agrees upon in terms of effort
  - Ideally, about half the team, about half the time
- Assign an arbitrary Fibonacci number to the Reference Story
- For each Backlog Item, assign the closest Fibonacci number that accurately reflects the effort ratio between that Item and the Reference Story
- Refine your estimates by continually comparing newly-estimated Backlog Items to previously estimated Items to ensure consistency in the effort ratios

*Don’t spend more than a few minutes estimating each Item in the belief that you will get better estimates!*
How Can This Possibly Work?

“As the number of samples increases, the sum of estimation errors approaches zero.”
– The Law of Large Numbers

- The implicit range ‘built-in’ to the Fibonacci Series ensures that some estimates will be over, others will be under

- Given enough Backlog Items, while each estimate may be off by +/- 60%, the sum of the estimates will be very close to the total effort represented by the Backlog
Example:
Story Points For Travel Backlog

- Fly non-stop from Los Angeles to New York City: 3
- Fly non-stop from Los Angeles to Sydney: 8
- Fly non-stop around the world at the equator: 21
- Fly non-stop to the moon via ‘magic’ airliner: 233
Getting to Yes

- **Estimate via consensus**, not coercion

- **Planning Poker (Wideband Delphi-like)**
  - Discuss the Backlog Item
  - Repeat:
    - Everyone prepares a story point estimate individually and privately (perhaps from their ‘story point card deck’)
    - Everyone reveals their estimate simultaneously
    - Everyone discusses any differences
  - Until:
    - Everyone agrees on the estimate
Use Historical Data If Possible!

Basing estimates on historical data is the most accurate way to estimate

- Use reference stories from previously-completed Product Backlogs
- Use velocity data from previously-completed projects by the same team
  - Less precise if from other teams in the same organization

Another reason to keep teams together across projects…
Estimating Velocity From Scratch

- **Start with an estimated Product Backlog**, if possible

- Create a tentative iteration plan by having the team ‘SWAG’ what they could do during a sprint
  - Do this at several places throughout the Product Backlog
  - Be mindful of the Definition of Done!

- Average the points per iteration in the plan to get the upper bounds (Expected Velocity), then divide by 2 to get the lower bounds (Worst-Case Velocity)
  - Be willing to lower the Worst-Case Velocity, especially when working with newly-formed teams and introducing Scrum!
Example:

Project Schedule/Plan Inputs

- **Project Scope**: 520 points
- **MRF Scope**: 378 points

- **Worst-Case Velocity**: 20 points/sprint
- **Best-Case Velocity**: 42 points/sprint

- **Sprint Duration**: 2 weeks
- **Project Team**: 8 workers @ $2.5k/week
Deriving Duration

- **Best-Case:** MRF Scope / Best-Case Velocity
  - 378 points / 42 points per sprint = 9 sprints to completion (18 calendar weeks)

- **Worst-Case:** Project Scope / Worst-Case Velocity
  - 520 points / 20 points per sprint = 26 sprints to completion (52 calendar weeks)

- **Initial Duration Prediction:** MRF Scope / Worst-Case Velocity
  - 378 points / 20 points per sprint = 19 sprints (38 calendar weeks)
Project Schedule/Plan Outputs

- **Project Duration**
  - **Don’t forget** to consider holidays, vacation, etc.!
  - Shorten schedules by using multiple teams, if possible

- **Project Cost**
  - Best Case: 9 * $40K = $360K
  - Worst Case: 26 * $40K = $1.04MM
  - Initial Prediction: 19 * $40K = $760K
Remember The Cone!

- **Commit to MRF** by the calculated dates
- Recalculate Project data after each sprint
  - Based upon Last Sprint Velocity
  - Based upon 3-Sprint Moving Average of Velocity

- **Base commitments on demonstrated performance**, not initial estimates
  - A reasonable commitment needs at least 3 sprints’ worth of velocity data
  - Spend less Planning Phase time planning, and run those sprints to get historical data on your current project

- Use the Sprint Retrospective to increase velocity throughout the project
7 Steps to Project Success

1. Work with your customers to understand what they actually want
2. **Create accurate effort-based estimates**
3. **Make realistic plans based upon estimates, make realistic commitments based upon demonstrated performance**
4. Empower your teams
5. Focus on delivering the highest value as early as possible
6. Do one thing, do it well, and finish what you start
7. Obtain constant feedback on progress, product, process… and act upon it
Questions?
Construx Software is committed to helping individuals and organizations improve their software development practices. For information about our training and consulting services, contact info@construx.com.

Seminar Schedule: www.construx.com/calendar