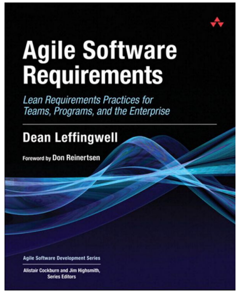
CE Department

Software Requirements Engineering

40688

These slides are designed to accompany Agile Software Requirements (2011) by Dean Leffingwell and support the university course Software Requirements Engineering, instructed by Mehran Rivadeh. Created and designed by Mahnaz Rasekhi.


Agile Software Requirements
Lean Requirements Practices for Teams, Programs, and the Enterprise
Dean Leffingwell
Foreword by Don Reinertsen
Agile Software Development Series
Alistair Cockburn and Jim Highsmith, Series Editors

Agile Software Requirements (2011)
Dean Leffingwell

Agile Estimating And Velocity

Chapter 8

Mehran Rivadeh
mrivadeh@sharif.edu
Software Requirements Engineering
October 2025 - Fall 1404 - SUT

Contents

1. Introduction

Introduction

- ➔ The **traditional** project **estimating means** never actually worked for software projects.
  - ◆ Use a work breakdown structure to identify every last task
  - ◆ Estimate each task
  - ◆ Add the tasks up
  - ◆ Build a Gantt Chart
  - ◆ Predict the cost and schedule

Introduction

- Everyone wants more reliable estimates. Otherwise, there is no way to predict how much value a team can deliver in a time period.
- If an agile team can reliably predict what it can do in a short timebox, then we can more reliably predict what a few teams working together can deliver in the next month or two.
- Even though this sounds like a small step, it actually achieves two major advances that executives will find very exciting:
    - Know where you are now
    - More accurately predict where you will be next

Introduction

- **Know where you are now**
  - The agile project knows exactly where it is at every point in time because it is based on a subjective evaluation of working code (really working, as in coded, integrated, tested, evaluated, and running).
- **More accurately predict where you will be next**
  - With effective agile estimating, teams have a fairly reliable predictor of what will be working in the next few weeks.
  - They know what is working now, and they can predict what they will deliver next based on the velocity they have achieved in prior iterations.

Contents

2. The Business Value of Estimating

The Business Value of Estimating

- ➔ As compared to coding and testing at least, estimating is overhead.
- ➔ Some might even look at estimating as a form of waste.
- ➔ In some kanban implementations, teams don’t bother much at all with estimating.

So, why do we bother to do estimating at all?

Estimating provides substantial value added for several reasons:

- ➔ **Determining Cost**
  - ◆ Effort is our proxy for cost. If we can’t estimate effort, we can’t estimate cost.
  - ◆ If we have no way to estimate cost, then we have no reasonable basis for going about our business at all.

The Business Value of Estimating

- **Establishing Prioritization**
  - Development effort predicts time, which is in turn a primary factor in cost of delay (CoD).
- **Scheduling And Commitment**
  - It is unreasonable to ask a team to commit to delivering an unknown thing in a certain time frame, especially longer term.
  - However, gaining commitment to near-term deliverables is important because it materially affects the planning and business objectives of our customers.
  - It also impacts our internal customers—those who document, train, deploy, and support the system.

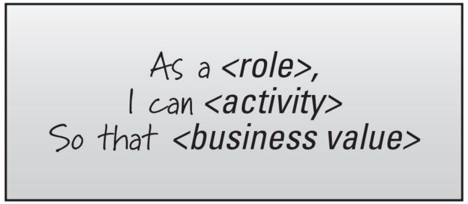
Contents

3. Estimating Scope With Story Points

Estimating Scope With Story Point

How do teams estimate the work they do in an iteration?

- ➔ Fortunately, we have small, independent "objects" to estimate—the user story.
- ➔ We know they deliver value because we designed them that way.
- ➔ If a team can estimate stories, then they can estimate:
  - ◆ The things of value they can deliver to a customer.
  - ◆ When they can deliver it.
- ➔ The most common method is the art of **relative estimating with story points.**



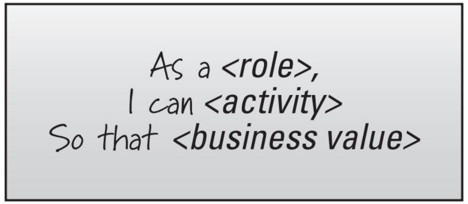
As a *<role>*,
I can *<activity>*
So that *<business value>*

Estimating Scope With Story Point

Story Point

➔ A story point is **an integer number** that represents an aggregation of a number of aspects, each of which contributes to the potential "bigness" of a story.

- **Knowledge**: Do we understand what the story does?
- **Complexity**: How hard is it to implement?
- **Volume**: How much of it is there? How long is it likely to take?
- **Uncertainty**: What isn't known, and how might that affect our estimate?



As a <role>,
I can <activity>
So that <business value>

Contents

4. Understanding Story Points

Understanding Story Points

Relative Estimating

Exercise: Assign “dog points” to each of the following types of dog to compare their bigness.

Understanding Story Points

Relative Estimating

Did you come across the questions below?

- What does the instructor mean by **bigness**?
  - Height, weight, mass, muscle, bite, attitude?
- What the heck **kind of poodle** is it? Standard poodle? Toy poodle?
  - "Hey, it makes a big difference!"
- What **scale** should we use?

Understanding Story Points

Relative Estimating

- ➔ You were struggling with **ambiguity**.
- ➔ There will always be ambiguity **in the estimating process**.
- ➔ When in doubt, **ask the product owner** for clarification.

Understanding Story Points

Estimating Poker

1. Participants include **all agile team members**.
2. **The product owner** participates but **does not estimate**.
3. Each estimator is given **a deck of cards with** 0, 1, 2, 3, 5, 8, 13, 20, 40, and 100 as their “**value**.”

Understanding Story Points

Estimating Poker

a. For each story, the product owner reads the description.

b. **Questions** are asked and answered.

c. **Each estimator privately selects a card** representing his or her estimate.

d. **All cards are simultaneously turned over** so that all participants can see each estimate.

e. **High** and **low estimators explain** their estimates.

f. After discussion, each estimator **reestimates**, and the cards are turned over for a second time.

g. **The estimates will likely converge**. If not, the process for that story is repeated until it does.

Repeat until all stories are estimated.

Understanding Story Points

Estimating Poker

Exercise 1:

1. As a student, I want to **estimate how many steps it would take to walk from our classroom to the cafeteria** so I can plan my break time.

Understanding Story Points

Estimating Poker

Exercise 2: Estimate within the timebox (maybe 10 minutes).

1. As a user, I want to **upload a profile picture to my account** so that others can recognize me and my profile feels more personalized.

2. As an admin, I want to **generate monthly usage reports** so that I can monitor system activity and share performance insights with stakeholders.

➔ Some amount of preliminary design discussion is appropriate. However, **spending too much time on design discussions is often wasted effort.**

Understanding Story Points

Subtle Aspect of Estimating Poker

1. It provides the best possible estimate based on **the collective judgment** of the team.
2. The range of numbers (0, 1, 2, 3, 5, 8, 13, 20, 40, 100) is cleverly designed.
   a. The **lower range** (0, 1, 2, 3, 5, 8) is designed to **help teams** more **precisely estimate small things** they understand well.
   b. **The gaps** in the sequence become larger as the size of the estimate increases, **reflecting greater uncertainty.**

Understanding Story Points

Subtle Aspect of Estimating Poker (Cont.)

3. The expanded range (20, 40, 100) at the end of the series indicates that even larger items have even greater uncertainty.
    a. The story is too big for an iteration.
    b. It probably represents a feature or epic.
    c. It represents substantial risk and needs to be split.
4. Zero gives the teams a way to ignore small stories that can be implemented in just a few hours.

Understanding Story Points

Subtle Aspect of Estimating Poker (Cont.)

5. A consensus must be achieved before a final estimate is reached.
6. Reducing estimation bias.
7. It happens pretty fast.
    a. If an iteration is two weeks, even one hour represents 1.25% of the total available time.

Understanding Story Points

How Much Time Should We Spend On Estimating?

- Spending too much time does not generally increase the accuracy of the estimates.
- It bores the team.
  - given too much time, the team may start to talk themselves out of the original estimates as the stories become seemingly more familiar.
- The first estimates were better because they reflected a healthier respect for the unknowns.
- More investment in estimating time rarely has a material effect on the actual estimates.

Understanding Story Points

How Much Time Should We Spend On Estimating?

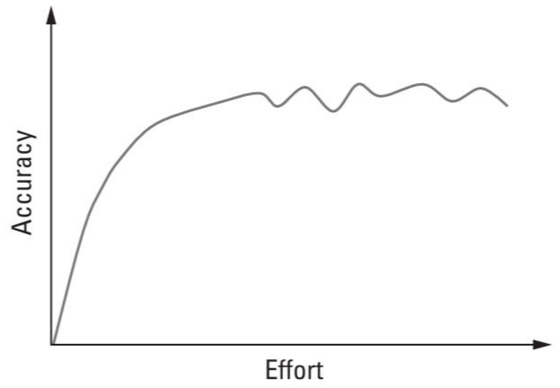


Figure 8–3 A little effort in estimating produces the most accurate results.

Understanding Story Points

What’s up with the estimates for “estimate the cubic volume of the room”?

Team 1	**Estimation**
Estimate cubic volume (within 30%).	5
Estimate cubic volume (within 30%).	8
Team 2	**Estimation**
Estimate cubic volume (within 30%).	4
Estimate cubic volume (within 30%).	9
Team 3	**Estimation**
Estimate cubic volume (within 30%).	6
Estimate cubic volume (within 30%).	40

Understanding Story Points

What’s up with the estimates for “estimate the cubic volume of the room”?

- ➔ Teams 1 and 2 were in a modest-sized, cubic conference room with low ceilings.
- ➔ Team 3 was in a much larger space with high vaulted ceilings and a very complex geometry.
- ➔ Without a ladder, it wasn’t even clear how they could come up with the 30% estimate, much less 5%.

Understanding Story Points

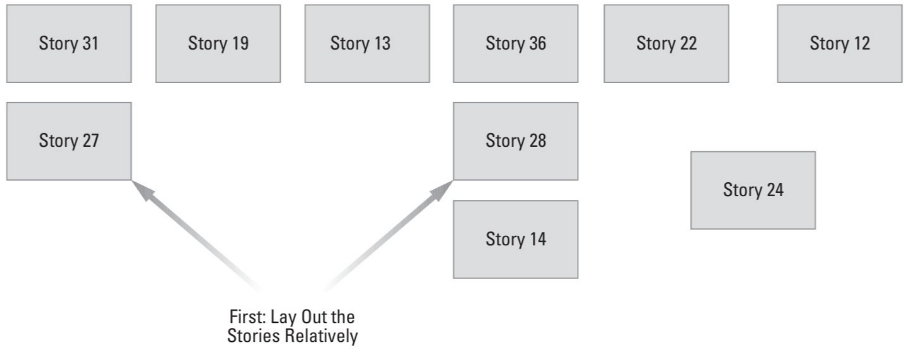
- Before you compare team estimates for theoretically comparable user stories, you must first understand **what kind of room** (software platform, programming languages, new team versus experienced team, computing resources, legacy versus green-field development, and so on) **each team is in.**

Contents

5. Tabletop Relative Estimation

Tabletop Relative Estimation

- Like planning poker, this technique involves the entire team but clearly requires face-to-face communication.
- The team discusses each story in the backlog and places the story on the table in a size position relative to other stories.
  - Small stories to the left
  - Bigger stories to the right



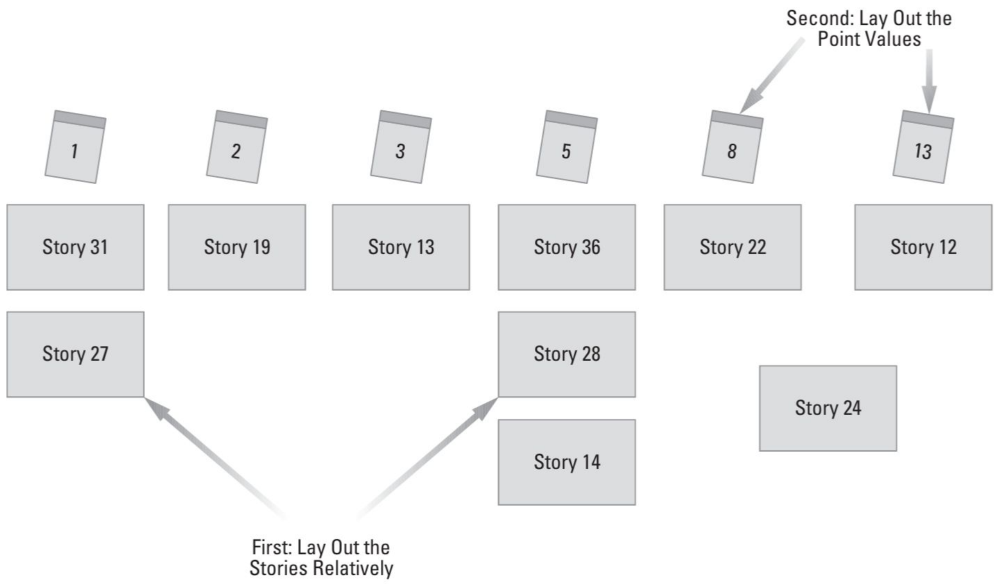
Tabletop Relative Estimation

Advantages

- ➔ As the stories are discussed, it’s likely that there will be some **second thoughts** and **shuffling of stories.**
- ➔ The process is **fast** and **visual**.
- ➔ Each story can be seen with respect to all the other stories.
- ➔ As the prospective iteration unfolds, **the team gains a sense of the work ahead**.

Tabletop Relative Estimation

- ➔ The stories aren't really estimated yet; they are just placed in relative sizes.
- ➔ To create the actual estimates, points can be assigned to columns.



Contents

6. Team Velocity

Team Velocity

How Long Will It Take The Team To Deliver Anything?

- To answer this question, we need to know **the team's velocity**.
- A team's velocity is simply **how many points that team can complete in a standard iteration.**
- Given a team's known historical velocity in a given domain,
    - They can now predict **how long it will take them** (how many iterations) **to complete an arbitrary amount of work.**
    - It is also **a valuable calibration** that will help them do the next round of stories more accurately.

Contents

7. Caveats Of Relative Estimation

Caveats On The Relative Estimating Model

- ➔ It is based on historical data and is predictive only to the extent that the future (new stories) looks like the past (stories already completed).
- ➔ It is valid only to the extent that the team continues to have the same individuals.
- ➔ A team’s velocity cannot be compared to any other team.

Caveats On The Relative Estimating Model

Be Careful What You Ask For

- ➔ The goal of every agile team is to continuously increase velocity while improving quality at the same time.
- ➔ Even though it is predictive and even though it measures the team's ability to achieve a certain amount of functionality in a time period, it is not a true (or at least complete) measure of productivity.
- ➔ As such, velocity can be a fairly reliable predictor of short-term future events, but it is not a tool for managing teams.
- ➔ It is only a tool by which teams manage and measure themselves.

Caveats On The Relative Estimating Model

Be Careful What You Ask For (Cont.)

- If management attempts to use velocity as a measure of team performance, the team will respond in one of three ways.
  - Continuously improve the team’s true productivity and agility in all aspects.
  - Cut back on quality, building technical debt for a future period.
  - Simply increase the size of the estimates.

Contents

8. Schedule And Cost Estimating

Schedule And Cost Estimating

Estimating Schedule

- ➔ If we know size and velocity, we can calculate how long it will take to do something.

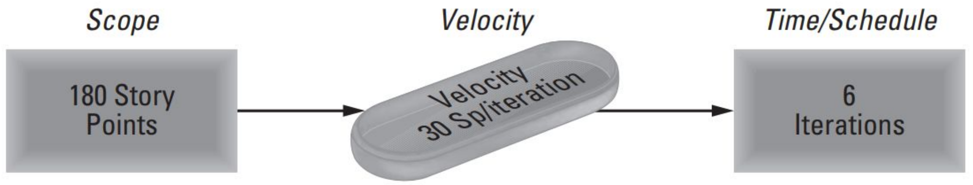


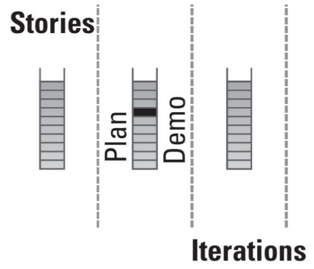
Figure 8–7 Converting story points to time

Schedule And Cost Estimating

Estimating Schedule

1. Teams estimate each individual story in the backlog they are trying to schedule.
2. Then they add those estimates together.
3. Since they also know the length of their iterations, they use the following simple formula to estimate how many days it might take to work an arbitrary backlog.

   # days to do the work = # days per iteration * (backlog size estimate / velocity)



Schedule And Cost Estimating

Estimating Cost

- ➔ Take the average burdened cost for a team and divide it by their velocity.
  - ◆ That provides the cost per story point for that team.
- ➔ Then when the team estimates an arbitrary backlog, just multiply the cost per story point for that team by the total estimate for the backlog.

Contents

9. Estimating With Ideal Developer Days

Estimating With Ideal Developers Days

- In IDDs **the story** is simply **estimated** based on **the number of total person days**, including development and test, that the team thinks they will need to accomplish the story.
- IDDs are conceptually **simpler than story points**.

Estimating With Ideal Developers Days

Problems With The Story Point Method

- ➔ It isn't so easy to understand by the team, and their outside stakeholders.
- ➔ It's hard to get started.
  - ◆ Until teams have done a few iterations, they have no idea how to predict what they can accomplish.
- ➔ Getting to schedule and cost estimates is very indirect.
- ➔ Teams occasionally struggle to adjust their velocity based on the availability of team members.
- ➔ Team velocities are not normalized.

Estimating With Ideal Developers Days

IDDs Technique

➔ The team looks at each story, discusses it with respect to the same complexity factors we described earlier.

➔ Then estimates how many IDDs it will take to do the story.

➔ The reason the estimates are called “**ideal**” developer days is that the team typically deprecates their capacity for planning, demos, management meetings, and other team and company overhead items.

Estimating With Ideal Developers Days

IDDs Advantages

- ➔ Teams have always done it that way.
- ➔ Management needs schedule and cost estimates anyway.
- ➔ It’s far easier to understand and explain.
- ➔ It’s easy to adjust velocity for sick leave, vacations, training, and so on.

Estimating With Ideal Developers Days

IDDs Serious Disadvantages

- ➔ Teams tend to get caught up when estimating in times.
- ➔ It’s far more personal and can be politically loaded.
- ➔ It’s the way we used to do it, and heaven knows, that didn’t work very well.

Contents

10. A Hybrid Model

A Hybrid Model

- Each team is guided to **estimate the smallest story** as a **1**.
    - Story that can be done by one person in about a day.
- Each team is also guided to initially estimate that **they have eight IDDs per team member per two-week iteration** (or adjust accordingly).
    - This leaves about 20% for planning, demoing, company functions, training, and other overhead.

A Hybrid Model

The Advantages

- ➔ The teams can still use planning poker, and gain most all those tangible benefits.
- ➔ The estimate is still a consensus, and it doesn’t say who is doing it.
- ➔ They can start immediately.
  - ◆ They have their first velocity estimate (8 * team members) on day one.
  - ◆ The relative methods still avoid any tendency to overinvest in estimating.
  - ◆ If your choices are only 1, 2, 3, 5, and 8, you can’t be breaking things down into hours.
  - ◆ So, it goes just as fast.

A Hybrid Model

The Advantages (Cont.)

- ➔ The Translation To Cost Is Obvious.
    - ◆ Average the daily cost across all practitioners, including burden.
    - ◆ The cost for one point is equal to that number, multiplied by 1.25 (because we also have to pay for the days that are not included in the IDD).
- ➔ Normalizing Velocity

End of Chapter 8

Contributions

- ➔ Author of Reference Book: **Dean Leffingwell**
- ➔ Course Instructor: **Mehran Rivadeh**
- ➔ Slide Creator: **Mahnaz Rasekhi**
  - ◆ These slides are primarily based on Agile Software Requirements by Dean Leffingwell, with occasional adaptations to enhance clarity and engagement.