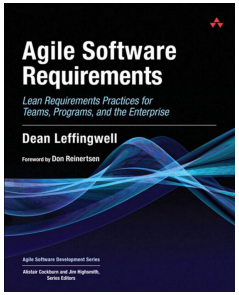
CE Department

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 (2011)
Dean Leffingwell

User Stories_Part 2

Chapter 6

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

INVEST In Good User Stories

Contents

INVEST In Good User Stories

- Writing the code for an understood objective is not necessarily the hardest part of software development; rather, it is understanding what the real objective for the code is.
- The attributes of a good user story.
  - **I**ndependent
  - **N**egotiable
  - **V**aluable
  - **E**stimable
  - **S**mall
  - **T**estable

Bill Wake

INVEST In Good User Stories

Contents

Independent

- Independence means that a story can be **developed**, **tested**, and potentially even **delivered on its own**. Therefore, it can also be independently **valued**.
- Many stories will have some natural sequential dependencies as the product functionality builds, and yet each piece can deliver value independently.

Independent

Sequential dependencies Vs. Independence Value

Feature	Dependency	Independent Value
Display a single record	None	Shows basic data
Display a list of records	Builds on single record	Lets users see multiple entries
Sort the list	Needs list display	Helps users organize data
Filter the list	Needs list display	Let’s users narrow down results
Paginate the list	Needs list display	Improve useability for large datasets
Export the list	Needs list display	Enables data sharing or backup
Edit items in the list	Needs list display	Allows data correction or updates

Independent

Why Independence Matters?

- ➔ **Flexibility:** Teams can prioritize and rearrange stories more easily.
- ➔ **Parallel Work:** Developers can work on different stories without blocking each other.
- ➔ **Incremental Delivery:** You can release useful features sooner, even if the full functionality isn't complete.
- ➔ **Better Estimation:** Independent stories are easier to size and plan.

Independent

Non-valued Dependencies → we need to find and eliminate

Example:

- ➔ As an administrator, I can set the consumer's password security rules so that users are required to create and retain secure passwords, keeping the system secure.
- ➔ As a consumer, I am required to follow the password security rules set by the administrator so that I can maintain high security to my account.

Independent

Non-valued Dependencies → we need to find and eliminate

Example (Cont.):

- ➔ The consumer story depends on the administrator story.
- ➔ The administrator story is testable only in setting, clearing, and preserving the policy, but it is not testable as enforced on the consumer.
- ➔ In addition, completing the administrator story does not leave the product in a potentially shippable state—therefore, it's not independently valuable.

Independent

Non-valued Dependencies → we need to find and eliminate

Example (Cont.):

- ➔ As an administrator, I can set the password expiration period so that users are forced to change their passwords periodically.
- ➔ As an administrator, I can set the password strength characteristics so that users are required to create difficult-to-hack passwords.

INVEST In Good User Stories

Contents

Negotiable

- Traditional requirements: a contract for specific functionality.
- A **user story** is a **placeholder** for **requirements** to be **discussed**, **developed**, **tested**, and **accepted**.
- This process of negotiation between the business and the team recognizes the legitimacy and primacy of the business inputs but allows for **discovery through collaboration** and **feedback**.
- **Agile** is founded on the concept that **a team-based approach** is **more effective** at **solving problems** in a **dynamic collaborative environment**.

Negotiable

- A user story is **real-time** and **structured** to **leverage** effective and direct **communication** and **collaboration** approach.
- The **negotiability** of user stories **helps** teams achieve **predictability**.
- The **lack of** overly constraining and **too-detailed requirements** enhances the team's and business's ability to make **trade-offs** between **functionality** and **delivery dates**.
- Because each story has flexibility, the team has more **flexibility to meet release objectives**, which increases dependability and fosters trust.

INVEST In Good User Stories

Contents

Valuable

- An agile team's goal is simple:
  - To deliver **the most value** given their **existing time** and **resource constraints.**
- Therefore, **Value** is the most important attribute in the IN**V**EST model.
  - Every user story must provide some value to the user, customer, or stakeholder of the product.
  - Backlogs are prioritized by value.
  - Businesses succeed or fail based on the value the teams can deliver.

Valuable

- A typical challenge facing teams is
  - learning how to write small, incremental user stories that can effectively deliver value.
- Traditional approaches: Functional breakdown structures based on technical components
- This **technical layering** approach to building software **delays the value delivery** until all the layers are brought together after multiple iterations!

Valuable

Bill Wake:

> Think of a whole story as a multi-layer cake, e.g., a network layer, a persistence layer, a logic layer, and a presentation layer. When we split a story [horizontally], we're serving up only part of that cake. We want to give the customer the essence of the whole cake, and the best way is to slice vertically through the layers. Developers often have an inclination to work on only one layer at a time (and get it "right"); but a full database layer (for example) has little value to the customer if there's no presentation layer.

Valuable

- To create valuable stories, reorient functional breakdown structures:
  - From a horizontal → To a vertical approach
- We create stories that slice through the architecture so that we can present value to the user and seek their feedback as early and often as possible.

Valuable

- Although normally **the value is focused on the user interacting with the system**, sometimes the value is more appropriately focused on a customer representative or key stakeholder.

Example: Improving **the click-through rate on ads** - On the website

- **From the perspective of the end user:**

  As a consumer, I can see other energy pricing programs that appeal to me so that I can enroll in a program that better suits my lifestyle.

- **From the marketing director's perspective:**

  As a utility marketing director, I can present users with new pricing programs so that they are more likely to continue purchasing energy from me.

Valuable

- ➔ Another challenge faced by teams is:
  - How to articulate value from technical stories? Such as:
    - Code refactoring
    - Component upgrades, and so on.

Example:

- ➔ How would the product owner determine the value of the following?
    - Refactor the error logging system.

Valuable

Example:

- As a consumer, I can receive a consistent and clear error message anywhere in the product so that I know how to address the issue.

OR

- As a technical support member, I want the user to receive a consistent and clear message anywhere in the application so they can fix the issue without calling support.

- The **value is clear** to the user, to the product owner, to the stakeholders, and to the team.
- Articulating the value of a technical solution as a user story will **help communicate** to the business its relative importance.

INVEST In Good User Stories

Contents

Estimable

- Although a story of any size can be in the backlog, in order for it to be developed and tested in an iteration, the team should be able to provide an approximate estimation of its **complexity** and **amount of work** required to complete it.
- The **minimal investment in estimation** is to determine whether it can **be completed** within **a single iteration**.
- Additional estimation accuracy will increase the team's predictability.

Estimable

- ➔ If the team is unable to estimate a user story, it generally indicates that the story is:
  - **Too large**
    - If it is too large to estimate, it should be split into smaller stories.
  - **Uncertain**
    - If the story is too uncertain to estimate, then a **technical** or **functional** spike story can be used to reduce uncertainty so that one or more estimable user stories result.

Will be discussed …

Estimable

- One of the primary benefits of estimating user stories is not simply to derive a precise size but rather to draw out **any hidden assumptions** and **missing acceptance criteria**.
- To clarify the team's shared understanding of the story.
- The conversation surrounding the estimation process is as (or more) important than the actual estimate.
- The ability to estimate a user story is highly influenced by the size of the story.

INVEST In Good User Stories

Contents

Small

- ➔ User stories should be small enough **to be able to be completed in an iteration**.
- ➔ Otherwise, they can't provide any value or be considered done at that point.
- ➔ Smaller user stories provide **more agility** and **productivity** because
  - **Increased Throughput**
    - Smaller stories mean the team can complete more of them in a sprint.
    - This gives faster feedback and more visible progress.
  - **Decreased Complexity**
    - Smaller stories are easier to understand, implement, test, and review.
    - Less complexity means fewer bugs and smoother collaboration.

Small

Increased Throughput

- ➔ Little's law:
  - ◆ **Cycle Time = Work In Process / Throughput**
    - Cycle Time = The time elapsed between the beginning and end of the process.
    - Work in Process = The amount of things we are working on.
    - Throughput = The amount of work that can be done in a unit of time.
  - ◆ In a stable system, throughput is constant.
  - ◆ We have to decrease work in process in order to decrease cycle time.
- ➔ Highway example …

Small

Decreased Complexity

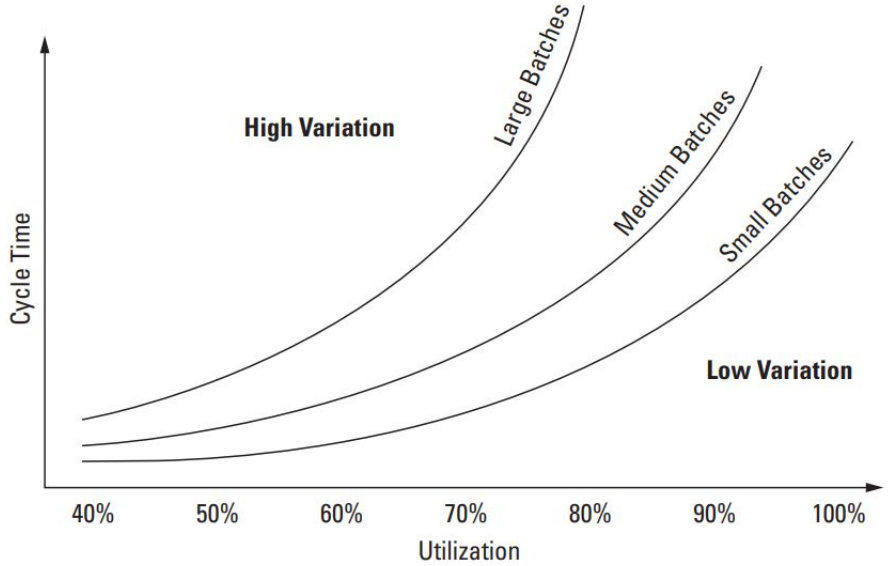
- ➔ Smaller stories move faster of their decreased complexity.
- ➔ Complexity has a nonlinear relationship to size.
- ➔ This is seen most readily in testing, where the permutations of tests required to validate the functionality increase at an exponential rate with the complexity of the function itself.
- ➔ This is one of the primary reasons that the **Fibonacci estimating sequence** (that is, 1, 2, 3, 5, 8, 13, 21 ...) is so effective in estimating user stories.
    - ◆ The effort estimate grows nonlinearly with increasing story size.

Small

On The Relationship Of Size And Independence

- ➔ It seems logical that smaller stories increase the number of dependencies.
- ➔ Smaller stories, even with some increased dependency, **deliver higher value throughput** and **provide faster user feedback** than larger stories.
- ➔ So, the agilist always leans to smaller stories and then makes them smaller still.

Small


Large Batches
High Variation
Medium Batches
Small Batches
Low Variation
Cycle Time
40%
50%
60%
70%
80%
90%
100%
Utilization

INVEST In Good User Stories

Contents

Testable

- In proper agile, all code is tested code.
- It follows that stories must be testable.
- If a story does not appear to be testable, then the story is probably
  - Ill-formed
  - Overly complex
  - Dependent on other stories in the backlog.

Testable

Write The Test First

- To assure that **stories don't get into an iteration if they can't get out** (be successfully tested), many agile teams today take a "write-the-test-first" approach.
- This started in the XP community using Test-Driven Development (**TDD**), a practice of **writing automated unit tests prior** to **writing** the **code** to **pass** the **test**.
- Since then, this **philosophy** of **approach** is being **applied** to **development** of **story acceptance criteria** and the necessary **functional tests prior** to **coding** the **story itself**.
- If a **team really knows how to test a story,** then **they likely know how to code it** as well.

Testable

- To assure testability, user stories share some common testability pitfalls with requirements.
- Vague words such as **quickly**, **manage**, **nice**, **clean**, and so on, are **easy** to **write** but very **difficult** to **test** because they mean different things to different people and therefore should be avoided.
- And although these words do provide negotiability, **framing them with some clear boundaries** will help the team and the business share expectations of the output and avoid big surprises.

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.