

---

**First Edition**

---

# **Introduction To User Stories**

---

**Kevin O'Mara, O'Mara Consulting Associates.**

# Overview

---

Software requirements is a communication problem. User Stories represent customer requirements rather than document them.



# What Is a User Story?

A User Story is one or more sentences in the language of the end user that captures what the user wants to achieve. User Stories are frequently used with **Agile** software development methodologies. Each User Story is limited, and should fit on a 3x5 card to ensure that it does not grow too large. User Stories should be written by or for the customers of a software project and are their main instrument to influence the development of the software.

User Stories are a quick way of handling customer requirements without having to create formalized requirement documents and without performing administrative tasks related to maintaining them. The intention of the User Story is to be able to respond faster and with less overhead to rapidly changing real-world requirements.

User Stories are composed of three aspects:

- written description of the story used for planning and as a reminder

- conversations about the story that serve to flesh out the details of the story
- tests that convey and document details and that can be used to determine when a story is complete

Ron Jeffries named these aspects Card, Conversation, and Confirmation.

An initial User Story can be as simple as “*The Marketing Manager can upload documents to the content management system so they can be shared with the sales force.*”

## Where are the details?

Clearly the simple User Story above doesn't provide enough detail for the developers to start coding and testing.

Many details can be expressed as additional stories. It is better to have more stories than to have stories that are too large. Big stories are often referred to as **epics**. At the start of a project almost all the stories will be epics that capture the “big ideas” of the project. Eventually epics will be de-composed into User Story sized functionality - something that can be implemented in a half day to perhaps two weeks.

In addition, details come from the conversations about the story. That information can be added as simple notes on the



Story Card (or its electronic equivalent). But the better way is to capture those details as Acceptance Tests.

**FIGURE 1.1** User Story on a 3x5 card

172: ADD ITEM TO BASKET

AS A customer

I WANT to add items to a basket

SO THAT I can review them later

PRIORITY: 8

ESTIMATE: 13

This format follows Mike Cohn story model.

- upload PDF, video, and Excel documents
- upload a zero length file (should fail)
- upload very large files
- upload a file that already exists

## What Are Acceptance Tests?

Acceptance testing is the process of determining if the software performs as the customer expects. Referencing our User Story “the Marketing Manager can upload documents to the CMS”, **Acceptance Tests** might be:



# Writing Stories

---

2

INDEPENDENT

NEGOTIABLE

VALUABLE

ESTIMABLE

SMALL

TESTABLE

INVEST



# The INVEST Model

Bill Wake, author of *Extreme Programming Explored* and *Refactoring Workbook*, suggests six attributes of a good story:

- Independent
- Negotiable
- Valuable (to users or customers)
- Estimable
- Small
- Testable

## Independent

As much as possible, care should be taken to avoid introducing dependencies between stories. At the time you are writing the stories, you don't know what business priority they may have - and therefore the order in which they may be implemented.

## Negotiable

Stories are short descriptions of functionality, the details of which are to be negotiated in a conversation between the customer and the development team. They are reminders to have that conversation.

The result of those conversations and negotiations are annotated on the story card and/or written down as Acceptance Tests.

## Valuable

User Stories are expressed in terms that provide some value to the users of the software - keeping in mind, as we'll discuss in the next chapter, there are several User Roles to represent.

The key thing is to avoid writing stories that are only valued by developers.

Instead of: The system must scale to support 10,000 users.

Say: The Sales Manager wants to sell 10,000 licenses.

Developers will understand the second statement means they have to architect the system to scale to 10,000 users. Expressing it in business value terms facilitates prioritizing it according to its value to the business.



## Estimable

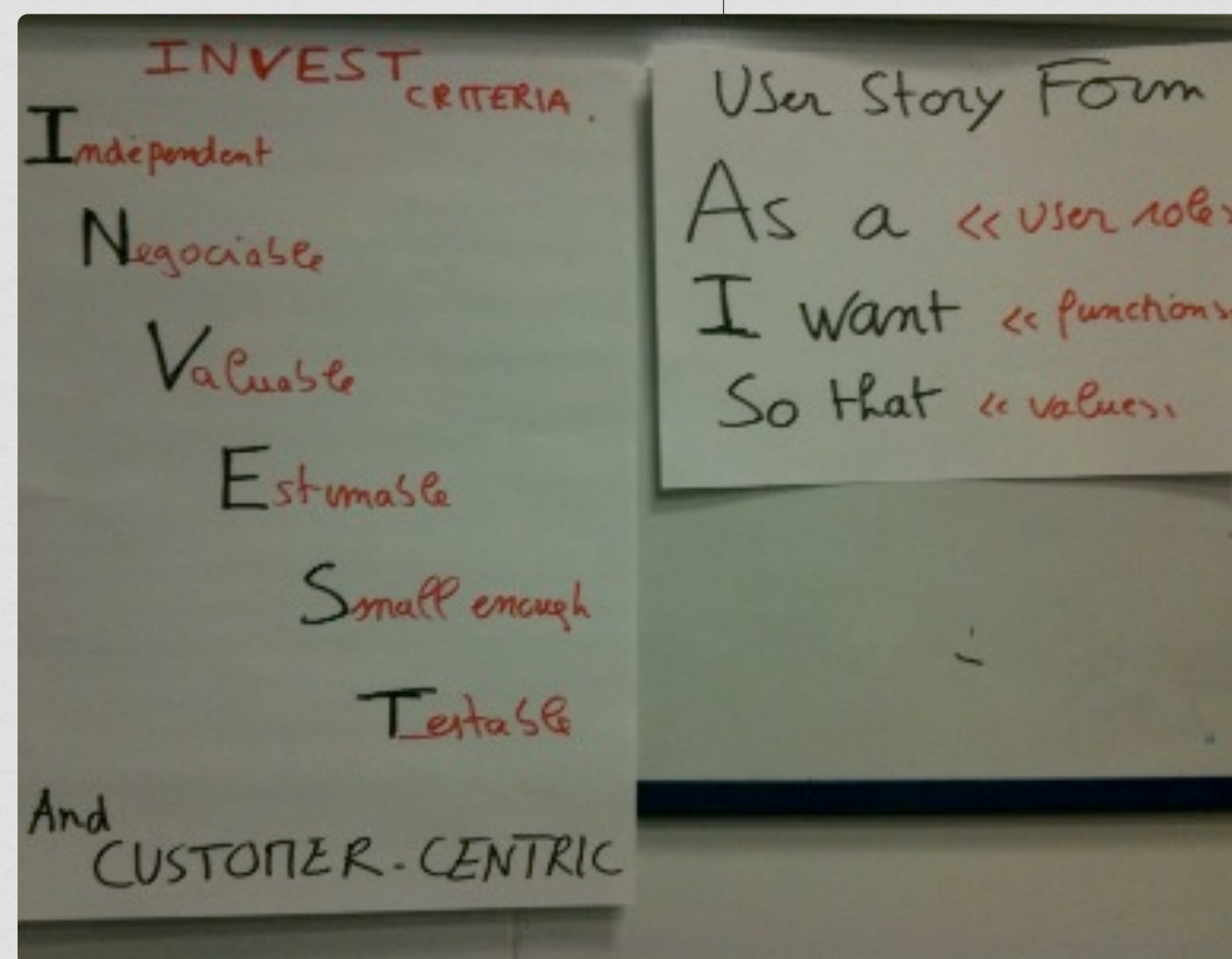
To manage the project it must be known how much time it will take to turn a story into working software. There are three common reasons why a story may not be estimable:

1. Developers lack domain knowledge
2. Developers lack technology knowledge
3. The story is too big

While User Stories should be written in the language of the users, taken to the extreme this is just as bad as writing requirements in technical language. Care should be taken to either avoid or explain domain jargon

- and of course domain knowledge will be transmitted to the developers in conversations with the customer.

FIGURE 2.1 The INVEST model



If the developers lack knowledge about the technology involved, it can be handled by what Extreme Programming calls a Research Spike - a brief experiment to learn just enough to be able to estimate the troublesome story. In this way the story turns into two stories - a quick research spike and then the real work.

## Small

As mentioned earlier, a User Story should be implementable in a short amount of time - somewhere between a half day and two weeks. Smaller stories waste administrative time and larger stories are difficult to estimate with any accuracy.

There are several strategies to break down large stories (epics)

---

into smaller ones. Epics are typically either compound or complex stories.

A compound story is actually composed of several smaller stories - for example “the Marketing Manager can upload documents to the CMS” is actually a compound story. After discussing the story with the customer or **Product Owner**, we discover there is also meta-data about the documents that must be added to the CMS, and there is a need to determine which end users have access to specific documents. So our simple epic needs to be decomposed into its parts.

Complex stories are inherently large and cannot be easily disaggregated into a set of constituent stories. Often complex stories have uncertainty associated with them, and one strategy is to split the story into an investigative story and a developing story.

Splitting stories that are too large into “right sized” stories can be challenging, but it is imperative it be done.

## Testable

User Stories must be written so the software can be tested to determine if the story has been completed. If the story seems untestable it may be a system constraint - for example, “the user must find the software easy to use” is actually a general constraint on the system.



# User Role Modeling

---

# 3

Writing User Stories as though there is only one type of user can lead to missing stories for users who do not fit the general mold. User Centered Design teaches us the benefits of identifying user roles and personas.



# User Roles

Understanding the broad range of users at the early stages of designing software is important. Brainstorming the breadth of users is helpful when determining the scope of a project as it often exposes features that are derived or afterthoughts. The idea of the administrative role or the stewardship role are often overlooked when focus is on the first, usually customer focused, release.

## Why user roles?

Being able to tell stories from many different perspectives, all within each iteration of new features, creates a focused and working solution within each iteration. Having a broad set of user roles identified keeps the design comprehensive and the features current.

Role	Who
Marketing Manager	Tina
Sr. Sales Rep	Bob
System Administrator	Ryan
CFO	Scott

hensive and the features current.

Many teams extend the role concept to Personas, borrowed from User Centered Design. Personas add more information to the role in the form of personal characteristics about the typical person in that role - or making it more clear there are a wide range of characteristics of people in the role - e.g., the notion of novice versus power user.



# Role Modeling Steps

The process of role modeling involves four steps:

- brainstorming an initial set of user roles
- organizing the initial set
- consolidating roles
- refining the roles

Participants in Role Modeling should include the customer and as many of the product developers as possible.

## Brainstorming

Each participant grabs a stack of 3x5 cards. Start with everyone writing role names on cards and placing them on a table.

When a new role card is placed, the author says the name of the role and nothing more - there is no discussion or evaluation of the roles. Each person writes as many roles as they can

think of. There are no turns - each person just writes a card whenever they think of a role..

This continues until progress stalls and participants are having a hard time thinking up new roles. Rarely does this need to last longer than 15 minutes.

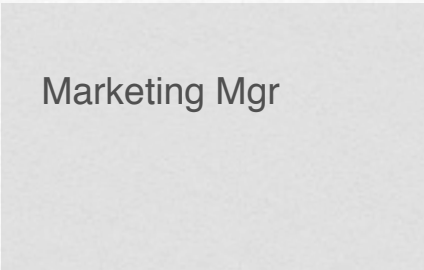
## Organizing

Once the initial set of roles are identified, the cards are moved around on the table so their positions indicate the relationship between roles. Overlapping roles are placed so their cards overlap entirely. For example:





---



Marketing Mgr

## Consolidating

After the roles have been grouped, the participants consolidate and condense the roles. The authors of overlapping cards describe what they meant by those roles names. If equivalent, the roles can be consolidated.

In addition, the participants should evaluate whether certain roles are unimportant to the success of the system and eliminate those roles. And of course the discussion may reveal new roles that had not been previously identified.

After consolidation, the remaining roles are arranged on the table to show their relationship to each other.

## Refining

At this stage the roles can be refined by adding key attributes to each role - i.e., create Personas from the roles. The attributes should provide useful information to the developers - for example:

- The frequency with which the user will use the software
- The user's level of expertise with the domain
- The user's general level of proficiency with the target technology
- The user's level of proficiency with the software being developed or modified
- The user's general goal for using the software - e.g., convenience, rich experience, ease of use.



# Story Workshop

---

The Story Workshop is a cross-functional meeting during which the participants write as many stories as they can.

4



# Story Writing Steps

Many descriptions of the software development processes refer to “gathering” requirements as if they were pieces of fruit that grow on trees. Agile processes acknowledge that it is impossible to get all the requirements in one pass, and also that there is a time dimension to stories. The relevance of stories changes based on time and what stories are added to the product in prior iterations.

Story details will also be added over time as well. In an Agile process like Scrum, the stories near the top of the Product Backlog will have more detail than stories toward the bottom, and stories taken into the Sprint will be discussed in detail during the Sprint Planning phase and throughout the Sprint. It must be understood that the estimates on the stories further down the backlog (the ones with fewer details), have more uncertainty.

At the beginning of each release of a product, it is worthwhile to conduct a Story Writing Workshop to identify as many of the big idea stories as possible.

## Low Fidelity Prototype

A good story-writing workshop combines the best elements of brainstorming with low fidelity prototyping. The prototype is built up iteratively on cards, paper, or whiteboard and maps very high level interactions within the planned software. The participants brainstorm the things a user might want to do at various points while using the applications.

The idea is not to identify actual screens and fields, instead the focus is on conceptual workflows.

Referring to Figure 4.1 for example, each box represents a component in the product. The component’s title is underlined in the box, with a few bullets about what the component does. Arrows between the boxes represent possible flow based on user interaction.

Start the prototype by deciding which User Role you’d like to start with - you’ll repeat the process using each role. Draw an empty box and ask what the selected user can do from here. For each action, draw a new box with a link from its originator.

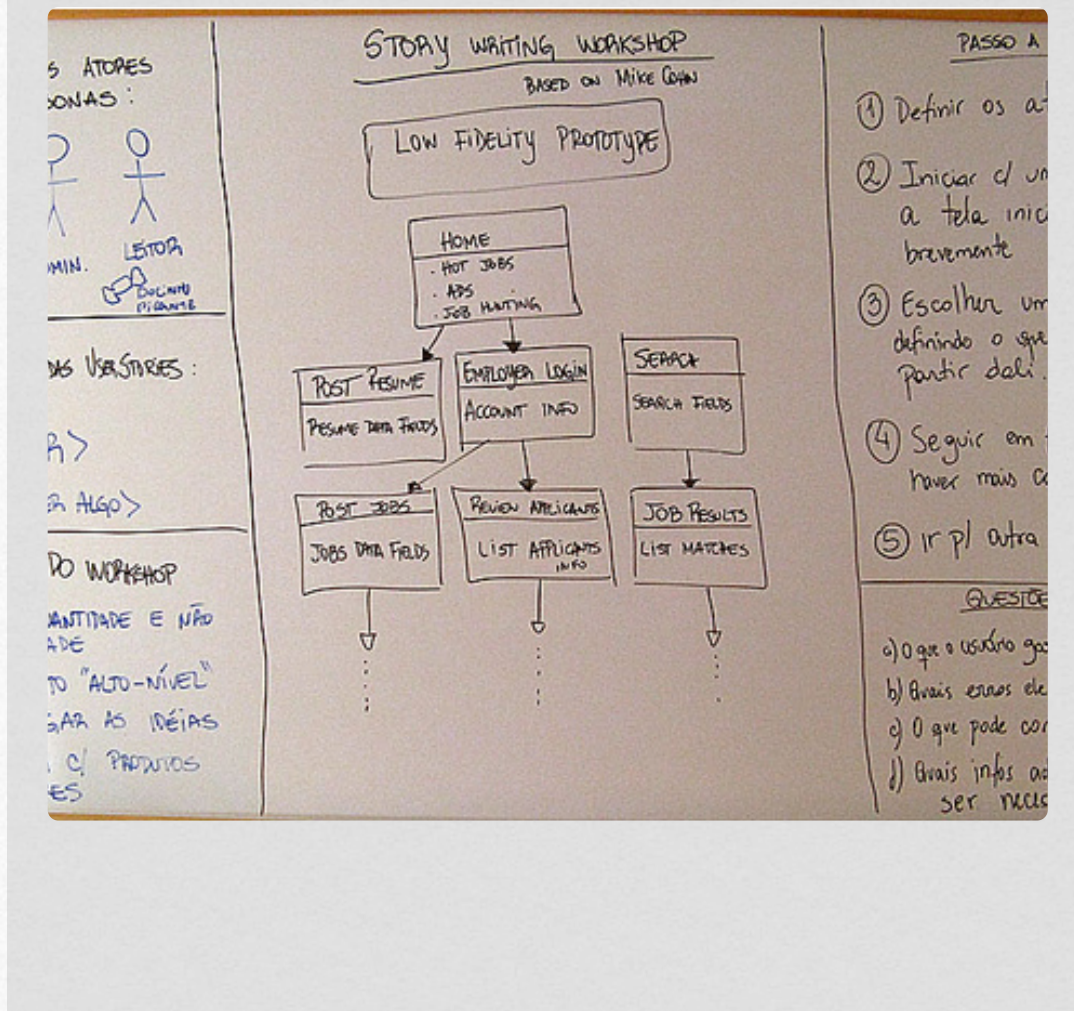
## Generate Stories

As the discussion continues write down the User Stories on 3x5 cards. As this stage you only need to capture the big idea.



None of the discussion at this stage requires knowledge of user interface design or system architecture. More importantly, dis-

**FIGURE 4.1** Low Fidelity Prototype



cussion about these items is counter-productive.

The goal is to generate as many stories as possible in a short amount of time. Again, more details will be added over time.

## Estimating

The last quarter of the workshop should be reserved for the Team to provide effort estimates for each of the stories. Often some of the participants in the workshop will be Subject Matter Experts (SMEs), customers, and or end-user. While it may be advantages to those participants to be present during the estimating process, they are only there to answer questions the Team may have to help clarify the stories - they absolutely do NOT provide input relative to the actual estimates.

## Estimating Story Points

In an Agile process like Scrum, the Stories are estimated in "Points" which completely arbitrary. The important thing is to ensure that Points are normalized across all the stories - in particular, if story A is worth 3 points and story B is worth 6, the Team is saying they think story B will take twice as much effort (time) to implement.

Another recommendation is to use **Fibonacci** numbers for Points. As Fibonacci numbers get large fairly quickly, those numbers help reflect the uncertainty of big stories - if the Team is really unclear what and how big the feature is, it gets an appropriately large number.



# User Story Template

Mike Cohn's template for a User Story is:

As a <<type of user>>,

I want <<some goal>>

so that <<some business value>>

## The Template

Mike Cohn's template epitomizes the elegance of the User Story. In one sentence it identifies the User Role, some feature required of the software, and the business benefit that will accrue.

It serves as an excellent starting point and basis for the Conversion and Confirmation aspects to come.

As the example illustrates, the first part of the template can be used to capture the initial story. As time passes the developers annotate the story and add Acceptance Tests.

Each row of the Acceptance Test portion of the template represents a test. Each test is expressed in terms of a GIVEN starting condition, a user ACTION, and the RESULTing condition.

In our example, the first test answers the question "what happens if the user tries to upload a document that has already been uploaded."

Template:

	<<story name>>
AS A	<<user role>>
I WANT	<<some goal>>
SO THAT	<<some reason>>

Example:

	Document Upload
AS A	Marketing Manager
I WANT	To upload documents to the Content Management System
SO THAT	I can share them with the sales force

Acceptance Test Template:

BEGINNING STATE	ACTION	RESULT

Acceptance Test Example:

GIVEN	ACTION	RESULT
Document already exists	Upload attempted	Error message displayed
Any	Zero length document upload attempted	Error message displayed
Any	PowerPoint upload attempted	Progress bar displayed, success indicated with message



# Acceptance Test

Acceptance testing is the process of determining if the software performs the way the customer expects it to work.

Software developers often distinguish acceptance testing by the system provider from acceptance testing by the customer (the user or client) prior to accepting transfer of ownership. In the case of software, acceptance testing performed by the customer is known as user acceptance testing (UAT), end-user testing, site (acceptance) testing, or field (acceptance) testing.

---

## Related Glossary Terms

Drag related terms here

---

## Index

Find Term

Chapter 1 - What Is a User Story?

Chapter 2 - The INVEST Model

# Agile

Agile software development is a group of software development methodologies based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change. It is a conceptual framework that promotes foreseen interactions throughout the development cycle. The Agile Manifesto introduced the term in 2001.

---

## Related Glossary Terms

Drag related terms here

---

## Index

Find Term

## Chapter 1 - What Is a User Story?

Chapter 4 - Story Writing Steps



# Epic

An epic is a large user story. There’s no magic threshold at which we call a particular story an epic. It just means “big user story.” At some point in the project an Epic will be broken down into User Stories that are small enough to implement.

---

## Related Glossary Terms

Drag related terms here

---

## Index

Find Term

### Chapter 1 - What Is a User Story?

Chapter 1 - What Is a User Story?

Chapter 1 - What Is a User Story?

Chapter 2 - The INVEST Model

Chapter 2 - The INVEST Model

Chapter 2 - The INVEST Model



# Fibonacci

In mathematics, the Fibonacci numbers are the numbers in the following integer sequence:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

By definition, the first two numbers in the Fibonacci sequence are 0 and 1, and each subsequent number is the sum of the previous two.

Fibonacci sequences appear in biological settings, such as branching in trees, arrangement of leaves on a stem, the fruitlets of a pineapple, the flowering of artichoke, an uncurling fern and the arrangement of a pine cone.

---

## Related Glossary Terms

Drag related terms here

---

Index



# Product Owner

The Product Owner represents the voice of the customer and is accountable for ensuring that the product delivers value to the business. The Product Owner writes customer-centric items (typically user stories), prioritizes them, and adds them to the product backlog.

---

## Related Glossary Terms

Drag related terms here

---

Index