Table of Contents

How Do We Know We Have a Good User Story? The INVEST Criteria

How do we know if we have a “good” User Story? People use the mnemonic to make sure they have good Stories to work with – INVEST. INVEST stands for:

- **I**ndependant: Work item is modular and can de delivered with no (or at least manageable) dependencies.
- **N**egotiable: A basic solution is articulated, with room for options.
- **V**aluable: The value is clearly articulated. This is defined in the “so that” part of the User Story.
- **E**stimable: Provides just enough information so it can be sized compared to other work (relative sizing).
- **S**mall: Small enough to be completed in 1 Iteration (Sprint). Some call this attribute “sized to fit”.
- **T**estable: Acceptance criteria has been developed and are understood by the Team.

The initial draft of a Story will not have all these characteristics. Rather the mnemonic guides the discussion to improve our understanding of the Story. Many teams establish a Definition of Ready (DoR) crtieria for a Story. In other words we are “READY” when the Story has the INVEST characteristics. Then as Story bubbles up the Backlog and it becomes a candidate for work, they refine the Story so that it has these characteristics helping to increase overall understanding.

Detail

OK, great, but what does this really mean?

(I)ndependant

Independant means that as the Story becomes something we want to work on, as it gets closer to the top of the Backlog, we want ensure the story is set up so we are able take the story on immediately without have to work a number of dependencies. This means we need to write Stories that are complete in their own right, that are a vertical slice through the application so they can be implemented a piece at a time.

Sometimes Stories have to be implemented in a particular order to deliver the value. If you write the Stories as a series of changes to existing functionalities, you can sequence the Stories in the order that allows you to implement the capability. If you find you often have dependencies with your Stories, this might be pointing to a more systematic issue that may need work.

Independant does not mean you have to remove the dependency, but you should work to minimize the impact. For example if your work involves working through third party systems integrators you will have to ensure that you collaborate with these third parties to ensure your plans align.

(N)egotiable

Unlike standard requirements in a scope document, Stories are really a record of a request for something to be done. They are not contracts. Stories that have too much detail are often a reaction to the Agile approach.

People often feel better because a lot detail gives the impression that people know what they are going to do. Agile says the best approach in a lot of cases is “try something and get feedback”. Stories are therefore negotiable and individual Stories will be updated, added to, new Stories created based on what we have learned and our understanding of the overall Goals of what we are trying to achieve.

The idea behind “negotiable” takes this one step further. It states that a basic solution is articulated, but there is still plenty of room for options. By ensuring we have a basic approach defined, we know the result is possible. Options will engage the collective intelligence of the Team to come up with the best approach.

Contrary to expectation that more detail is required, this flexibility will allow us to better meet our Goals.

(V)aluable

Stories are written from the perspective of the end user of the system. Stories are written in the “voice of the customer”. The value described here is the value the customer is expected to see when the capability has been implemented.

Many traditional development shops actually do not understand who the end user is and why they want it, and this is a problem. The idea is to really understand who wants this and why they want it.

(E)stimable

In most cases storys are used to plan and track the work we are doing, to provide information to the business so they can forecast what might happen. This means that we must be able to estimate a Story. We do this by estimating how big the work is, and then determining how much a Team can get done per Iteration (Sprint).

To have a estimable item, we need to have an understanding of the requirement as well as the approach we expect to take to implement. If there is a (big) problem either in understanding the requirement or the technical aspects then we will need some level of research before we start the working on the Story. This research is called a “spike” in Agile - a set of “discovery” work in this Iteration (Sprint) that gets us ready to a future (next) Iteration (Sprint).

The other problem with Stories that would stop us from estimating it is that perhaps the story is just too big. If this is the case, then we need to split the Story into smaller parts. This leads us to the next component of the characteristics of a good story – it has to be …

(S)mall

Stories need to be small enough to fit into an Iteration (Sprint). I've seen some people refer to the “S” as “Sized to Fit” rather than “Small”. Actually they need to be smaller than that. Typically you do 5-10 Stories to an Iteration (Sprint). If you have a typical 2 week Iteration (Sprint) this means the Team will need to close a story every few days in an ideal world.

If you remember that User Stories represent end user value you might be thinking “wow, we cannot deliver anything in 2 weeks, let alone 3 days”. This is a new skill you will develop as you do agile and like any useful skill you will take some time to develop it. To understand more about splitting stories see How Do We Split a User Story?

(T)estable

We need to understand how we know we have finished the capability – what tests will we run to confirm we have completed the work and provided the value.

What this means practically is that every Story we write should have an Acceptance Criteria and we should have clear, testable tests. We need to be specific so that, for example, we don’t say “feature should be fast / responsive” but perhaps say something like “response time is no more than 2 seconds under normal load”. As much as possible you want the acceptance criteria to reflect real examples of what you expect to see.

Want to Know More?

- How Do We Use a Definition of Ready (DoR)?
- How Do We Split a User Story?

FAQ, DIY, Agile, Mnemonic, UserStory, Ready, DoR

From:
https://www.hanssamios.com/dokuwiki/ - **Hans Samios' Personal Lean-Agile Knowledge Base**

Permanent link:
https://www.hanssamios.com/dokuwiki/how_do_we_know_we_have_a_good_user_story_the_invest_criteria

Last update: **2022/02/23 07:03**