**INVEST in User Stories**

**(adapted from DZONE article)**

**Independent**

Independency between stories is fundamental for being able to prioritize, add and remove stories from a release or an iteration plan as single units. Stories should be atomic, so that can be started and finished in isolation from other ones (like a database transaction).

Usually this completeness is achieved by defining a story as a **vertical slice** of an application, a feature that encompasses the database layer, the domain model and the user interface at the same time.

**Negotiable**

The original definition of story is that of a reminder to have a discussion with the customer/domain expert/customer proxy about this feature. A 20-page specification document that specifies everything is not a story.

**The story has to capture the essence of that has to be done, by whom, and why it has to be done. Everything that is not needed for prioritization and scheduling of iterations is not required initially**: it would be premature and would add weight to the story, making us less likely to drop it without fears when it is the case, due to the work already done on it.

**Valuable**

What this story is for? It is really needed? Moreover, it is **valuable to the customer or customer proxy** we are dealing with, or only to the programmers?

Adding fields to a database table without showing them is not a story, since it does not have value for the customer in itself and maybe a worthless task. The modifications needed to the database must be included in a valuable story (or many of them).

**Estimable**

As humans, we are not good at dealing with a range of values (estimation values) that span more than a magnitude. Typically stories are estimated in points or ideal hours, where the value range along the start of the Fibonacci sequence.

Sometimes a story comes up that would have a 0.5 or a 100 length. When a story is too big, you should divide it into simpler stories, or into a spike story of exploration with a fixed time limit plus another one to estimate after the spike.

Analogously, when a story is too short, you may bundle it up with other small stories or include it in one of the other ones. Not only **the size of a story influences estimation**: **also the understanding of the developers** on what is needed to complete the story. This is why the team in its entirety should estimate in a single seat.

**Small**

**The more stories are near to start the development phase, the more they should be kept small**, even by breaking up them in independent pieces. The 'Small' trait is related to the Estimable one: smaller stories are easy to manage, estimate and compose.

In fact, it is much more likely for a story to be too big then too small.

**Testable**

Acceptance tests are usually written on the back of the card (when using a physical system). The tests are written in a natural language definition, since code wouldn't fit on a card.

**If a story isn't testable, rethink it, or you'll never have a definition of *Done* on its development.** Done is when all the acceptance tests pass (hopefully automated ones).