A USER STORY IS JUST A FANCY NAME FOR An SRS?
BY RENEE TROUGHTON

In a query that recently spammed the Agile universe by Abder-Rahman Ali was a question posed about the difference between User Stories and Software Requirements Specifications, specifically

1) Do you think that “user story” is just a fancy name for SRS?
2) How do you compare a “user story” to “SRS”?
3) Do you think that “user stories” replace “SRS”?

4) Which of the two do you prefer working with?

So let me open with a clear positioning stance:

> *User Stories are not a fancy name for an SRS, they are not the same thing, they should replace an SRS when working in an Agile environment and I prefer working with User Stories.*

The following comments are focused on how I believe they are different and are naturally based on subjective opinion.

Origination

A SRS is usually created part way through a project, once the Business Requirement Specification (BRS) has been crafted and signed off. An SRS is crafted by a Business Analyst, often with minimal collaboration to other team members and the business throughout its development. As part of its development the BRS is cross referenced to ensure traceability. Once produced the SRS undergoes a review and approval process that does generally have an extensive audience. Often this review and approval process can take several versions until the document is deemed correct.

A User Story can be created at the start of the project or throughout the life of the project. Through a collaborative workshop(s) called “Inception”, features are defined. Often these features are broken down into User Stories using the Story Mapping technique. As everyone who needs to deliver the project is in the room there is no handoff of documents and no extended review and approval process.

The key differences in the origination process is that User Stories are not dependent on the existence of a BRS to be created, that everyone is engaged to identify and define the requirements, and that there is no formalized approval process (because it is not needed if we are one team, all responsible for the outcomes that we agree to).

A simple analogy is that Inception is requirements gathering on steroids.

Content

SRS content is often lengthy, comprehensive but lacking depth to implement against. Normally, an additional deliverable is produced in between the SRS being delivered and software starting coding (normally referred to as a Detailed Design Document, or DDD for short). Field rules, expectations of behavior under error or abnormal conditions are also specified within (sometimes as an appendix, or separated out into an additional deliverable). There are often multiple SRSs to a single BRS, but rarely would this number exceed a magnitude of 10. How a SRS is tested comes down to separate test deliverables.

A User Story is a promise for a conversation. It is often a single sentence written as a narrative of "As a <persona>, I want <requirement not solution>, so that <value obtained>". In addition to this single sentence the conditions under which the User Story will be tested are included as Acceptance Criteria. Well-formed User Stories, like system requirements, crosscut components or architectural tiers of the system(s).

Critically there is an expectation that the documentation of the User Story itself is unlikely to be sufficient or comprehensive. The conversation placeholder is the opportunity to discover these elements of the requirements and to clarify needs in person with a business representative. This conversation has representatives from all specialists. This conversation may also extend to a group of developers in order to determine the best architecture or design to deliver the User Story. In this respect a DDD is not normally produced, but photos of whiteboard outcomes are kept in a repository for prosperity. Field rules are often discussed as part of this conversation.

Expectations of behavior or abnormal conditions are often split out from the story – ie a new User Story is created per condition or behavior to handle expectations. In this respect a single business need, described within Inception as a Feature, may result in potentially several dozen User Stories. This is especially true when good User Story splitting principles begins to focus on one Acceptance Test per User Story and the team moves towards an estimation less environment of reduced User Story size variability.

Change

As part of the SRS development, as the BRS may undergo several change requests. As part of the DDD development, test deliverable development, actual development & testing the SRS may also undergo several change requests. Over the life of a Project this could result in hundreds of change requests to approved deliverables – this is to ensure that what is delivered is of value to the end user, otherwise the system is unlikely to meet their needs. Each one of these change requests takes the team away from delivering to estimate the impact of the change.

As User Stories never undergo an approval process, they can be changed at any time prior to a developer picking it up to deliver. The cost of change at this point in time is negligible with conversations often being the only highlight. Once development has begun there an team internal change policy often guides what happens next. As a rule of thumb, if the change would take the team less than 30 minutes to do then it should be absorbed, otherwise the change should be raised as a new User Story. That User Story then gets prioritized by the Product Owner and may

or may not be done based upon that priority. As a rule of thumb, unless the original User Story was completely off base, then it would still be delivered as it stands, otherwise it would be pulled from development.

Realizing value

An SRS does not have it's value ever realized or assessed. The BRS is tied to benefits, and it consequently gets assessed for benefits realization. Only through traceability of requirements is there confidence that the SRS realizes value. From a delivery standpoint, only once all the SRS(s) and BRS has been delivered does the software often go into production. The outcome is often a delivery near the realm of a year or more. The benefits realization activity is consequently months, if not years after the production delivery.

Both the Features and the User Stories realize value within their originating narrative of "so that…". As developers deliver User Stories they can go into production once they have been tested against their acceptance criteria. For iteration-based team this often means fortnightly deployments to production and consequently a fortnightly realization of value. Sometimes teams may not release to production at the end of these iterations because a minimal viable product is still not achieved. Some teams, especially those that have focused on technical excellence through continuous integration and continuous delivery, can realize value several times a day through regular deployments to production.

When Agile teams take on Lean Startup practices a User Story does not get deemed completed or done until analytics have been gathered in production to prove or disprove the value realization.

A good analogy of Lean Startup as it applies to User Stories is "continuous micro benefits realization". In this respect, Agile teams have a higher confidence that they are reducing the #1 waste in software development – building the wrong thing, because they are constantly checking with real customers if they are on the right path (not the proxy customer who may be the Product Owner or Business Stakeholder). The other key difference when Lean Startup applies to User Stories is that the narrative alters to no longer be a requirement and instead is a hypothesis that critically should be proven.

Estimation

When a BRS is completed the service delivery team does a detailed estimate to deliver not only the SRS, but all artefacts and actual software delivery outcomes. The change request process is often the only means to recover further money. Critically, this estimate is done when the largest number of unknowns are still unknown. Re-estimation after the SRS is completed is not normally done, especially in fixed price bidding environments. Estimates are commonly done by Project Managers and Architects and have next to no input from the people who do the work. There is no relationship to SRS and actual delivery rates (with the exception of the rare teams that still do Function Point Counting).

In Agile teams' estimation is done throughout. Both Features and User Stories are often estimated, but critically not in hours or days. Instead, Agile teams used comparative based

estimation techniques where one User Story's complexity is compared to other already existing estimated User Stories to best gauge its approximate size. As teams begin to deliver User Stories they track throughput rates, either by cycle time or amount of work done within an Iteration. This information is then used to forward project confidence of delivery. All team members are actively engaged and encourage to estimate the User Stories as a single unit.

Conclusion

It is hard for me to write all the above content and not continue to think "Why do we still do waterfall?" especially in the commercially fast paced world that we live in these days. An SRS is not the same thing as a User Story. The outcomes that each achieves are quite different because of how they are utilized, not because of the description of the need that is within them.