Smart Books

Scrum - Story Splitting

How can a Product Backlog Item, a "story" be decomposed?

Anne Due Broberg & Kurt B. Nielsen
Scrummaster.dk, info@Scrummaster.dk
Fyrre Allé 3, DK-6040 Egtved, Danmark

Version: 2014-03-26

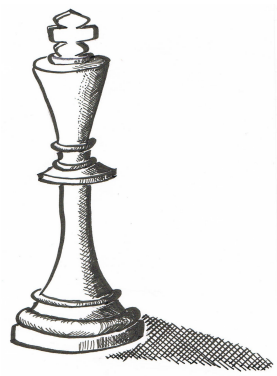
Table of Content

Document ID	WP-02-001	Filename	SmartBooks Scrum - Story Splitting.odt
Author	Kurt Nielsen	Revised	2014-03-26

1. Background for decomposing of Stories

Product Backlog Items (PBI) are the building blocks used to construct a Product Backlog representing the strategy of the project or initiative. PBIs are often described using narratives or “User stories”. There are many reasons to choose this pattern of specification.

The most important is that a story is easy for people with different backgrounds to understand. It is a perfectly normal sentence with ordinary words and grammar, it requires no special skills. Next is the fact that human being simply remember stories better than lists of facts.



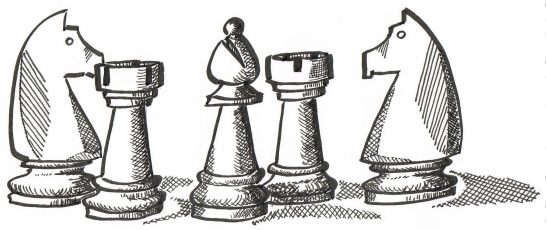
Product Owner

Stories are a sort of simplified language, enabling people from different domains of skills to be able to communicate, to work together on finding solutions to challenges and problems.

Often these stories are fairly large and coarse in the beginning, they are often referred to as “Epics”. There is a need to gradually refine these to gain understanding of details and be able to start implementing in iterations – sprints.

This is something that has to be learned and practiced. In the beginning it will seem hard to get your head around. A bit like watching birds you have to learn about the habits and nature of these creatures to be able to find and study them.

In this document we look at how Stories behave and what patterns and experiences there are to draw on, when trying to split them in smaller pieces that can be implemented with quality. When working with Scrum it is the Product Owner that have the top level responsibility for the whole body of Stories that constitute the Product Backlog – the specification of what we want to achieve with the initiative. He will however most of the time need help from the Team that actually does the implementation and others who are experts in the domain.



Team

1.1. Template for Stories

The idea of using Stories to write specifications, to formulate narratives of fundamental relationships between cause and effect goes back to 2001, whre a team at Connextra formulated the template that now is in frequent use[1]:

> *As such-and-such-a-user I would like to be able to do this-and-this-and-that in order to achieve ja-da-ja-da.*

A Story framed in this way is called a ”User Story”. It is clear who we are talking about, what they want and why. Especially the last part is very important. As it forces everybody to think about the value created by this deliverable, what the reason to do it is. This has made some people prefer the following variant:

> *In order to get ja-da-ja-da, such-and-such-a-user wants this-and-this-and-that.*

This variant has the benefit on focusing on the value bit right at the front.

Remembering the value part has some extra benefits: The Team will be able to challenge the this-and-this-and-that part, the solution, maybe they have a much simpler, more robust or cheaper way of achieving the same result. It is also very motivating for the Team to have a constant reminder of why this is important.

Often there is a need for further details of these Stories, to make sure that everything is covered. These details are usually called “Acceptance-criteria”, often there can be quite a long list of these (up to 15-20,

1 http://en.wikipedia.org/wiki/User_story

if there are more the Story should probably be split). The template for Acceptance Criteria is:

> *In such-and-such-a-situation, if event occurs, then the result must be such-and-such*
> *- or -*
> *Given a-certain-context and some-more-context. When this-event then such-an-outcome and another-outcome*

Often there will be a series og Global Acceptance Criteria that have to be fulfilled as well,these are often called "Global Constraints". A PBI or Story is not completed ("Done") until all Acceptance Criteria, global as well as local are fulfilled.

1.2. The INVEST principle

The following section take inspiration in an article by Bill Wake from 2003[2]. He introduces the acronym "INVEST" to explain what a good Story looks like:

I - Independent
N - Negotiable
V - Valuable
E - Estimable
S - Small
T - Testable

1.2.1. Independent

Stories are much easier to work with if we can concentrate on just the one story. We do not want them to overlap in concept, then we can execute them in any order. This is not always possible but a good principle to keep in mind.

1.2.2. Negotiable

A good Story can be negotiated. It is not a contract about features, but rather a reminder of details that have to be clarified between those who should use the deliverable (often represented by the Product Owner) and the Team delivering. A good Story captures the essence not the details, over time the Story will get notes, sketches and ideas for test, but these are not needed to prioritize or plan the Stories.

1.2.3. Valuable

A Story must have a value. It has to be a value for the customer or the user (the beneficiary). The Team and everybody else may have legitimate opinions about value, but in the end it is the customer that counts.

This is especially important when splitting Stories. Think of the story as a layered cake with lots of different ingredients. When we split the cake vertically then in every slice there is something from every layer, but still maintaining the essence of the whole cake. The Team often has a tendency to focus on each layer and "get it right", but a perfect layer is of little value to the customer.

1.2.4. Estimable

It is possible to estimate a good Story. We do not need an exact estimate, but enough to give the Product Owner enough information to prioritize the Stories. This feature of being estimable is partly depending on the Story being "negotiable", it is difficult to estimate a Story that we don't understand. There is also a relation to size, it is notoriously difficult to estimate large Stories. Finally it depends on the Team, what is their experience in this area? Sometimes the Team will have to break out a "spike" in this Sprint and spend some fixed time to gain insight and then be able to estimate decently for the next Sprint.

1.2.5. Small

Good Stories have a tendency to be small, they normally represent a few days to a weeks work for a single person. If a Story is larger than this experience tells us that it is hard to get our head around and we overlook details in estimation and some times in implementation. The Story text it self should also be short and concise, the solution should be open to discussion but the story itself should not be open to in-

2 http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/

terpretation.

1.2.6. Testable

Every good Story can be verified, "do we have it or don't we?", it should be possible to test it.The sooner a Team has a handle on Acceptance Criteria and ways to test, the more productive the get and they will be able to determine clearly if they accomplished what the Story was about.

If the Customer does not know how to verify the fulfillment of a Story, then we have an indication of the Story not being clear enough or the customer not really being able to see the value in the Story. It could of course also be that the customer just need help to understand the concept of testing.

A Team can treat non-functional requirements (such as performance and usability) as things that need to be tested. Figuring out how to come up with and do these tests will help the Team learn the true needs.

1.3. A Story has to be "Sufficiently Small"

How small is "sufficiently small"? The real answer is when everybody involved understands it and it can be implemented with solid done criteria in a Sprint.

It is common practice to use Estimation or Planning Poker to estimate Story size in Story Points or ideal man days. Again it is common practice to say that any Story estimated as 1, 2, 3, 5, 8 or 13, is a "Story", if it is larger (20, 40 or 100) then it is an "Epic", and needs further breakdown in order to implement. Many Teams regard 13 as borderline, they prefer to decompose that as well. If some Stories are estimated to e of size 0 or ½, they are often combined to one Story giving some identifiable value to the Customer. These small stories are often bugs or small annoyances discovered.

Another way to look at this question is to consider a Story to have a max size of 1/8 to 1/15 of the Team's Sprint capacity, the amount of work the Team can deliver per Sprint, often called the Team's Velocity. This is based on the experience that an optimal number of Stories in a Sprint is between 8 and 15, with a tendency to have more Stories in larger Sprints.

If the Team is estimating in Story Points (relative sizes), then they choose some reference stories to make this happen. If we focus on Stories of Medium size (typically estimated to 5), we know that we have to have 8 to 15 of those in a Sprint, that means that the Team's velocity would be from 40 to 75, not 10 and not 500.

Let us do a quick calculation: If the smallest Sprint is two weeks, the smallest Team 5 people, 10% of the Sprint time evaporates in meetings, the Team has 75% of their time available for real work (the rest disappears in organizational activities, surfing the Internet etc.) and there is 40 hours per week, then there will be 270 hours of real work in a Sprint. In this case the exchange rate between Story Points and hours be between 270/40 = 6.75 to 270/75 = 3.6.

A similar calculation with 4 weeks Sprints would yield 972/40 = 24.3 to 972/75 = 13.0. This is consistent with most Teams end up having a Story Point be between 1 to 3 ideal man days. All these numbers are only guidelines. It is not important to be in the middle of the range, but far outside is empirically not helpful. Either we have to small a resolution and hence to little detail, or to big a resolution, that either is not real or without importance.

2. A Road map for Story Splitting

Here is then finally a road map for decomposing Stories. The section draws on Richard Lawrence's "How to Split a User Story" from 2012[3] og Mike Cohn's chapter about Sotry decomposition from "Agile Estimation and Planning"[4], Additionally the authors have added some.

2.1. Prepare the original Story

1. Does the originale Story satisfy the INVEST principle (apart from being small perhaps)?
    1. **No**: Combine it with another Story or rephrase is some way so that it is a good Story (perhaps

3 http://www.richardlawrence.info/wp-content/uploads/2012/01/Story-Splitting-Flowchart.pdf
4 http://www.mountaingoatsoftware.com/books/agile-estimating-and-planning

large).

2. **Yes**: Is the size of the Story sufficiently small (1/8 to 1/15 of Sprint capacity)?
    1. **Yes**: The Story is ready.
    2. **No**: Continue – it is necessary with further decomposition.

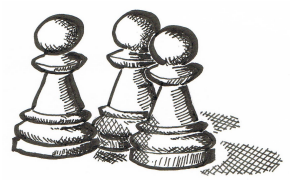
2.2. Use Patterns to decompose Stories

Here is a collection of simple patterns that can be used. The principle is: Look at the Story through the Patterns, does any of them seem obvious and improve clarity. There is no fixed solution, it has to "make sense", be possible to communicate to others.

2.2.1. Immediately visible fault lines

Are there obviously more stories hidden in the Story. Look for words like "and", "or", "then" and "unless". Also look for punctuation, full stops, dashes, parenthesis or other separators. All this indicates that fairly independent Stories can be formulated.

2.2.2. Split along different roles or "personas"



This pattern can be considered as a special case of the one above. If in the Story different roles appear they often do things slightly different, so basis for a split here.

2.2.3. Split Acceptance criteria as independant Stories

Often you can get an idea for decomposition by looking at the Acceptance criteria for a Story, some of these can perhaps be separated into their own Stories. The result is perhaps useful even before those final Acceptance criteria are fulfilled.

An example could be calculating of interest in a leap year. If there is still 3 years to the next leap year, we have the option of postponing that special case.

2.2.4. Workflow Steps

If the Story describes a workflow with different stages or perhaps more workflows, there are some possibilities here:

1. Is it possible to split the Story in more independent ones, each handling their own workflow?
2. Is it possible to split the Story so that the beginning and the end is implemented and then expand the solution with the middle part of the Workflow later?
3. Is it possible to take a thin slice of the work through the whole workflow and broaden later?

2.2.5. Variations in business logic

Does the Story have different business logic approaches built in? It is a variation of the workflow theme above. Take a part of the logic that make sense first and expand with the others later.

There is often a hint in this direction if there are expressions like "Flexible dates", "Different customer groups", "different subscription types" and so on.

2.2.6. Operations and actions

Does the story contain different operations or action being performed. In software solutions one often finds the fundamental CRUD (Create, Read, Update and Delete) operations. In general there are hints towards this pattern if words like "keep track of", "organize" or "configure", are found in the Story. These words often indicating something consisting of smaller actions.

2.2.7. Data variations

Does the Story do the same thing with different kind of items or data? Then try dealing with the different kinds of items or data one at a time. One recognizes this pattern if there are words like "both this and that" as the object of an action.

2.2.8. Interface variations

Does the Story have a complex interface or many sources of items or data? Is it complicated to activate.

1. Is there a simple variation that could prove the fundamental behavior?
2. Can the Story take input form one source first and the expand with the others later?

Often this pattern can be applied if the word ”or” is found, alternatively if it says ”from X, Y or Z”

2.2.9. Main effort

If the Story has multiple parts or variations, where the first one always will be the major effort to do and the rest simpler variations. Then choose the obvious, easiest, most valuable or riskiest part first.

This pattern can be used with different kinds of users, credit cards, screws or other elements.

2.2.10. Simple/complex

Does the Story have a simple nucleus that gives most of the value or the new knowledge? Then build this first and expand with the complicated bits later.

This looks a bit like “Main Effort” one above, but the focus here is on the big difference in complication. When using this pattern concentrate on getting value or knowledge quickly. The pattern is often useful when the Acceptance Criteria points to a lot of special cases.

2.2.11. Defer optimization

Is the story very big or complicated because it has lots of nun-functional requirements such as performance, beautiful icons or “cool” design? Then make it work first and make it pretty or fast later. It is important to get a result quickly.

Often this pattern can be used when there are requirement not directly related to the main function.

2.2.12. Last resort: make a timebox

Sometimes this is called “Break out a Spike”. If it is hard to find some decomposition that makes sense, there is no “visible fault lines”, then see if there is a small area that is clear enough to start with then carve this of as a Story.

Otherwise if it is possible to define which few questions (1-3) are preventing the Team from getting on with it, then define a “Spike story” set a time box (how much time should the Team spend on this maximum) with these questions, do it and reevaluate the Story.

If not it is time for a time out, seek help and try later.

3. Evaluate the decomposition

1. Are the new Stories within the same order of magnitude (1-13) and small enough to implement?
    1. **No**: Try again with another pattern on the original Story or the largest of the decomposed ones.
2. **Yes**: Does each of the new Stories satisfy the INVEST principle?
    1. **No**: Try another pattern.
3. **Yes**: Are there Stories that can be pushed down in Priority or deleted?
    1. **No**: Try another pattern, most likely there is some waste in some of the new Stories,
4. **Yes**: Is there an obvious Story to start with, that would give early value, knowledge or dreduction oin risk?
    1. **No**: Try another pattern to see if it can be achieved this way.
5. **Yes**: Done, if there is time you could try another pattern to see if youlike the result better.

4. Remember details during decomposition

Here are some extra things to keep an eye on during decomposition.

1. Do not throw the original Story away, leave it as an Epic (especially when using a Story Map), double check that everything is still there after the decomposition, and that on the other hand nothing unwanted in the form of extra requirements crept in.
2. When a Story is split consider where the business value goes. If splitting into 5 Stories what are the value of the new ones. The sum of the new 5 does not have to add up to the original estimate, but if it is far from it, there is something to discuss and investigate. Some times we cannot allocate a business value to the items broken down, it only makes sense for the customer if he has all of the original Story, we call this an MMF (Minimal Marketable Feature) then, it can still make sense to break down for implementation reasons, but don't force an unrealistic value estimate then.
3. The same with the estimate of effort. When breaking down it is quite common that the sum of the estimates goes up a bit, but on the other hand the uncertainty (risk) should go down.
4. Finally when splitting Stories, consider what happens with Acceptance Criteria. Step through them one by one and decide if it:
    1. Disappears, it has now become its own Story
    2. Follows onto one or more of the new Stories.
    3. In fact is common for all of the new Stories because it really belongs to the Epic level of the original Story.

Sometimes this all ends up seeming illogical, then try a new organization of acceptance criteria into common and local ones.