The Role of the Product Owner in Moving a Backlog Item to Done
(or It's Not Over Until the Fat Lady Sings)

Keywords: enabling specification, definition of done, empirical process, Scrum, acceptance criteria

Abstract: *This article explores how to achieve the productivity benefits of an upfront enabling specification, given the reality that Scrum is an empirical framework where emergent understanding of the story under development is inherent.*

I love reading the walls of team rooms. One team had a poster that read "The old guys have all the money." This was a reminder to the web team, staffed with youthful developers, to use large enough fonts and appropriate color schemes so that the most important segment of their target audience could actually read and use their web application without having to set the browser magnification at 150%.

Another team used the saying: "It's not over till the fat lady sings" to remind each other that no matter how well they think they have satisfied the written acceptance criteria, they are not done with a story until the Product Owner (PO) says they are done with it. Done is not some arbitrary state of a backlog item. When an item is "Done" users can take advantage of the new functionality and give meaningful user feedback. There is no sense in calling an item done unless from all points of view, including the Product Owner's, it is potentially shippable. Unfortunately, teams who adopt this philosophy to the extreme often face issues with erratic velocity and poor team morale. In my experience it is not uncommon for teams to struggle with how to deal with a PO's emerging understanding of a story when faced with an actual implementation. I'll illustrate with a couple of anecdotes from real recent projects.

> **Annoying UI**: The developer finished the story, validated it against all the acceptance criteria, and submitted it for testing. The tester explored what would happen if a certain form was filled in with a given field in error. The tester found that when the page was refreshed and the user prompted for correct data, the form was re-initialized to blank rather than simply blanking out the single field in error. The tester flagged this as a usability error and the PO required this "error" to be fixed before the story could be moved to "done".

> **Busy PO:** In this project, the team would regularly finish stories to the agreed upon definition of done, only to have the items sit in a queue waiting until the PO had the time to inspect and sign-off on them. This practice caused the team to have a very erratic velocity because often the PO did not review all completed items in time for them to be counted in the sprint in which they were done. This caused a team morale issue and led to the interesting behavior of the team sometimes "bribing" the PO to take the time to inspect and sign-off on completed items.

Suboptimal Functionality: The application is a class management system that allows students to upload homework, see their grades, read the class schedule, download the syllabus, etc. The story in question reads: "as an instructor I want to be able to give credit to a student for performing activities that do not result in a traditional homework submission so that I can motivate students to do these activities." For example the instructor might want to give credit to students for attending a cultural event or for filling out the class evaluation. The PO and entire team discussed how best to implement this story. The discussions were documented as a set of acceptance criteria and a GUI mock-up. The team then implemented the story and the PO sat down to validate the finished story. After running through the first several scenarios, the PO was quite happy with the implementation, but then, faced with the actual implementation, a scenario occurred to the PO for which the implemented solution was clearly sub-optimal if not infeasible. The PO was sorely tempted to move the story back into the in-progress column and require the team to redo the feature so that it would accommodate this newly envisioned scenario before allowing the story to be marked as done. The PO decided that this would not, however, be fair to the team. Instead, the PO wrote a new story to enhance, and rework part of, the already completed story. The new story was added to the backlog and the current story moved to done.

Discussion: In the case of "Annoying UI" centering on a blanked out form, the developer's position is that the estimate for the error handling for the story was based on the acceptance criteria as stated at the beginning of the sprint. To add to the acceptance criteria at the end of the sprint is not playing fair. Based on a concept introduced by Jeff Sutherland, the developer believes that the PO is responsible for developing an "enabling specification[1]" by the beginning of the sprint. Done should be measured by meeting a set of pre-agreed upon criteria. This philosophy certainly does facilitate maximizing velocity. It is very enabling for a team to know, at the beginning of a sprint, precisely what it is supposed to do and during the sprint just put their minds together and their heads down and get to it.

BUT, consider for a moment the origin of the term "enabling specification" This is a legal term that the US Patent and Trademark Office (USPTO) uses to describe the level of documentation that an inventor is required to produce to apply for a patent. The "enabling spec" must be written so that the average person knowledgeable in the domain can create the feature without having any discussion with the originators of the enabling specification.

In other words, an "enabling spec" is used to describe a known, refined process and resulting product that was in all likelihood discovered by the inventor using an iterative, creative, empirical approach. Sounds to me like "enabling spec" comes after the empirical work is done and goes with the subsequent plan driven approach used to replicate a known process. Scrum, an empirical approach, is being used precisely because we are building something that has never been built before. We don't know exactly what it is we want till we see it, and we aren't exactly sure how to build it. So why would we talk about using an enabling specification in an empirical process? One of the philosophical foundations of Scrum is

[1] http://scrum.jeffsutherland.com/2009/11/enabling-specifications-key-to-building.html

that when the business inspects, the development team may need to adapt. Thus many Scrum teams include the PO's personal sign off as a part of the definition of done.

However, there are many negative consequences to not having an upfront "enabling spec" for each backlog item. Both velocity and team morale can take a big hit. When a backlog item is not clearly specified, the development team cannot focus on rapidly developing the item. Furthermore if, in the team's mind, an item is ready to move to done because it meets all the acceptance criteria, but the PO moves it back to "in progress" because of the PO's emerging understanding of the item or the business, this can be very demoralizing to the team.

So, how can we account for the empirical nature of Scrum, but still have an "enabling spec" for backlog items? I find a 4-fold approach works best.

1. **The PO is responsible for advance grooming of the product backlog[2].** When a product backlog item (PBI) needs exploration, the PO should call upon whatever resources are necessary to explore all issues so that an "enabling spec" can be developed by the time the item is pulled into the Sprint backlog. The PO might work with subject matter experts, prototypers, the development team, usability experts, focus groups, or any other resources necessary. Sometimes these resources are pulled in on an ad hoc basis, and sometimes they form the basis for an ongoing PO "support team." Engagement of the PO with the entire team is essential.
2. **Plan some slack.** The team realizes and plans for, that in spite of the best product backlog grooming efforts, some requirements will only emerge during the engineering effort or when faced with the actual operational implementation. So team effort estimates take into account that a **limited** amount of re-work is anticipated based on PO feedback and emerging understanding. The Scrum Guide[3] recognizes this by calling for the establishment of a Sprint Goal[4] for each sprint. It is the Sprint Goal that is fixed for the duration of the sprint. Some requirements details within the scope of the sprint goal are expected to emerge and to be negotiated throughout the sprint.
3. **Continuous Requirements Elaboration and PO Engagement.** The team realizes that the initial acceptance criteria will need to be elaborated into a detailed set of test scenarios that elaborate not only business criteria, but also technical system properties including performance, scalability, security, and error handling. As the acceptance criteria are elaborated, new insights will emerge. To make sure developers are not surprised by these new insights requires an ongoing carefully choreographed interaction between the testers, developers, and PO. When I have been a PO, I require that for every PBI pulled into the sprint, an *acceptance criteria/test scenario development* task goes on the task board. As PO I remain engaged with the person doing that task, and personally sign off on the resulting test scenarios. Any other team member working on that PBI is careful to remain engaged with us.

---

[2] : https://sites.google.com/a/scrumplop.org/published-patterns/value-stream-pattern-language/product-backlog/enabling-specification

[3] The Scrum Guide is the "official" definition of scrum according to the founders and can be downloaded from: http://www.scrum.org/scrumguides/

[4] http://www.scrumalliance.org/articles/39-glossary-of-scrum-terms

A company in Chattanooga that recently implemented this suggestion claims that it is an important improvement to their Scrum process and that it has given their POs a chance to provide feedback even earlier in the development process. Reviewing the test scenarios as early as possible has helped them catch deficiencies in understanding sooner, resulting in less rework and improved productivity. For details of this technique please see the article "Dancing with Pigs.[5]"

4. **The PO Should Play Fair**. When a team delivers on an enabling spec, they deserve to have that story moved to done and get credit for it on the burn down chart. A good PO should only require minimal, reasonable, re-work during a sprint. If the backlog item "enabling spec" was wrong, incomplete, or ambiguous, the PO should take responsibility.

Anecdotes revisited: The case of the "annoying UI" highlights that an enabling spec should have two parts: one part specific to the backlog item, and another part that is a set of team or product standards common to all features. In this particular case, the team was missing a standard for how forms submitted with errors should he handled. The team's definition of done should be rigorous and include all requirements and standards, both functional and non-functional, so that the up-front estimates can take all these factors into account.

The case of the "Busy PO" above represents a serious dysfunction which the team must address. If the PO continues to neglect their responsibilities, and doesn't respond to the feedback from the scrum master and the rest of the team, it is time for the team to involve upper level management.

The case of "Suboptimal Functionality" might have benefited from more backlog grooming, team brainstorming, or prototyping. On the other hand this was likely a case where the team, including the PO, did the best they could. Then they inspected and found they needed to adapt, but the PO played fair and wrote up the adaptation as an additional backlog item, admitting that the team's results did handle most scenarios and did meet the agreed upon acceptance criteria.

Mature Teams: Officially the PO is the person responsible for approving the development team's implementation of the Sprint backlog items, but in a mature environment, the whole team helps groom the backlog, agrees that the specification is enabling, inspects what they produce and adapts as necessary.

Even though that in this article we have focused on the PO's role in moving a story to "done", It is usually suboptimal to draw too clear a line between the activities and responsibilities of the PO, the ScrumMaster, and the development team. They are all a part of the same Scrum team, and as the team matures, they function more and more as a single integrated organism where the left hand does, in fact, know what the right hand is doing. It is this tight integration of business and technology coupled with the continuous collaboration of mature teams that helps avoid surprises and end game rework.

So in Summary, it is not a question of "enabling specification" vs. empirical process. It is figuring out how to do "enabling specifications" within an empirical process. The product owner should ensure that

[5] http://www.scrumalliance.org/articles/364-dancing-with-pigs

whatever grooming work is necessary to provide the best enabling specification possible is done. However, because Scrum is an empirical process in which emerging requirements and understanding is inherent, sometimes the enabling specification will be wrong or incomplete. Plan some slack to allow for this, but if the amount of rework is substantial, treat the rework as a new backlog item.

As always, inspect and adapt!

Acknowledgements. I would like to thank the following reviewers for their insightful comments and suggestions for improvements. Roman Pichler, Clinton Keith, Kane Mar, Tom Mellor, Andreas Schliep, Michael James, Brian Rabon, and Jeff Southerland