Acceptance Test Driven Planning

Richard J. Watt¹ and David Leigh-Fellows²

¹ ThoughtWorks, UK
rwatt@thoughtworks.com
² Egg, UK
David.Fellows@Egg.com

Abstract. The experience of XP planning for many is not a successful one. We have found that by making acceptance tests not only central to the definition of a story but central to our process itself, they can be used to drive the entire development. This paper describes an adaptation, or evolution to XP style planning based around acceptance testing which takes the existing planning practices (with some additions) and organises them in a way that we believe can lead to better planning and more predictable results.

1 Introduction: How Do We Know When We Are Done?

One of the defining questions for a development team is “do we know when they are done?”. The question sounds simple but too often the answer is not as obvious as it should be. Acceptance tests are an effective way of expressing requirements in a way that provides an unambiguous answer to this question. As others have found, writing acceptance test definitions before coding begins on an iteration has both obvious and more subtle benefits. Given these benefits, why is it not more common for acceptance tests to be written before iteration planning begins? What are the obstacles and how can they be overcome? We have sought to find answers to these questions in the last year and in doing so have developed an adaptation of XP style iteration planning that has some key benefits over the traditional approach.

2 Iteration Planning: A Typical Experience

Our experience of iteration planning was perhaps typical. As a team fully signed up to the agile cause, we all agreed that this approach to planning was better than anything else we had tried. We also agreed that in our shared experience the process had never been wholly successful. Maybe we were doing something wrong but the more people we spoke to the more we realised our experience of planning was at least not uncommon.

As an illustration of our plight, below is a description of a typical iteration planning session:

We gathered all interested parties into a room which for us meant 10 developers, our customer, our QA engineers, an interaction designer, our project manager, iteration manager and coach – all empowered to contribute to the planning process.
Our customer would describe each of the stories in turn. The developers would ask questions and discuss solutions before producing a list of tasks for each story and an updated estimate of the effort it would take to implement the story. Unfortunately, this process of discussion often took a long time and was once described by our customer as feeling more like a “techno babble” session than a planning one. From the developers’ perspective, it was very difficult to produce confident estimates when the customer could not provide the information they needed in the meeting - this resulted in best-guess estimates on the incomplete information available. Even though our customer tried to consider all of the likely angles they couldn’t anticipate all of the questions that were going to be asked during the session. What seemed like a sensible approach to start with did not feel so good after a few painful, marathon planning meetings. The team was starting to lose confidence in the process.

The real pain came when we measured our velocity at the end of the iteration and discovered the discrepancy between what we had signed up for and what we had achieved. There are, of course, many reasons why a team could have a poorer than expected velocity but once we had investigated further we discovered that the main issue was that we hadn’t identified a complete set of tasks for each story. We could all see that the developers were working really hard and the results produced were good but in hindsight it was if they had started a 400m race but didn’t know where the finishing line was and just kept going round the track. Our understanding of the work was incomplete, which meant it was unclear when we were done. This also meant everyone was very tired and nobody was very happy - we knew we had to do something.

3 Getting Our Stories Straight

One of the biggest problems we had in our planning sessions was that our customer felt that no matter what they did to prepare there were still many questions they could not answer without time for further analysis and/or investigation. Our solution was simple: a day or two before the end of the iteration the customer would sit down with the QA engineer and a developer to begin the process of writing acceptance tests for the upcoming stories, or in our terms, “Getting our Stories Straight”. In short, we still ended up asking the same questions but by doing this activity before the planning session we were able to find more answers earlier and thus be better prepared for when we did gather the full team into a room. This is not a return to “big up front analysis” but an acknowledgement that as we approach the end of an iteration we already have a good idea of what is going to be completed in this iteration and the customer has a good idea of what stories they want to include in the next.

Our principal aim at this stage was simply to better prepare the customer for our planning session so that developers could get more of the answers they needed at the time they needed them, but we soon discovered there were many more benefits:

- **Are our tests any good?** Writing acceptance tests is a skill, our QA engineer helps our customer make sure the acceptance criteria specify the expected behavior and external quality[1]. One of our developers is on hand to make sure that there is no technical reason why the functionality required to pass the acceptance tests cannot be implemented. If we liken this to UML use cases, a story becomes the title of the use case and the acceptance tests become the use case itself detailing the main success scenario, alternate and error scenarios. [2].