ProviderSafe Analysis and Estimation

The challenge

The biggest obstacle with current practices in the ProviderSafe space relates to transparency and reportability. It's not enough to be productive; we also need to show in some measurable way *how* productive and provide some transparency into the flow of our work. While we don't necessarily work in the traditional software development model, we still have internal and external stakeholders who need to be able to understand the team's flow of work.

So what exactly is the challenge? The challenge is that Agile/Scrum tools such as Jira simply aren't designed to track and manage work in our current operational mode. A significant part of that relates to how we estimate tasks.[1] The other part is in how we define the work and break it down into manageable pieces (I say chunks, but that opens the door for people saying things like, "We're blowing chunks!").

Story Estimation

While I can understand why the practice of stories = days was adopted ("If we would go by the book, like Mr. Saavik, hours would seem like days."), that's not really how story points are supposed to be handled. Stories are estimated on a Fibonacci scale from 1-13.[2] A story point is a unit of complexity. It's intent is to answer the simple question "How complex is this task?" and by extension "How much effort is required to complete this task?"

Traditionally, this is used to estimate and understand the amount of work a team can complete in a given time frame. In Scrum that timebox is the Sprint, which may be set at anywhere from one week to one month.

Equifax has standardized on a 2-week sprint with the guidance being that 8 points is the maximum that should be assigned to a Story. In other words, an 8-point Story is one estimated with a level of complexity and effort meaning it will take an entire 2-week sprint to complete. If the team determines that a story is too complex to complete in a 2-week sprint, then it should be broken down into a series for Stories or Tasks that individually can be completed in the Sprint timebox and over time complete the request in its entirety.[3] (I will add that in traditional software development 13-point stories are used and represent the highest complexity for something completable in a Sprint. But P&E has said 8, so we'll use 8 as our frame of reference.)

Since we are operating in Kanban, we don't necessarily have a timebox in that sense, but we ought to nevertheless follow the P&E guidance of the 8-point scale and treat an 8 points as the maximum estimate for a story and consider it from the perspective of what is completable in a 2-week span.

Breaking Down Tasks

The corollary to the above is that we ought to invest some effort into determining how we can break down our work into smaller, more manageable, and, perhaps more importantly, more reportable stages. The question is how we can achieve that.

First, I think we need to establish a standard process to analyze the defined work and determine the steps or stages an item would go through from start to finish. The

analysis itself would be the first task in this process, the output of which would be the definition of the steps necessary to achieve the overarching objective of the story.

I think that means first of all that almost everything comes in to our project as a Story. The Story represents the full body of work that needs to be completed to have a releasable increment at the end of the cycle.

The Story would have multiple sub-tasks under it that represent the stages of work, the first of which would be the analysis task.

The remaining tasks would derive from the analysis and define what needs to be done step-by-step to satisfy the Story. I'm hoping that we can define tasks that break things down into stages that also establish a ballpark timeline with an expected completion date. That doesn't mean we are obligated to meet the expected date, but this at least gives us a target.

The breakdown for each story would hopefully show individual tasks with their own timelines. Stage 1: Do this, approximately *x* days of work; stage 2: do this, approximately *y* days of work, etc.

---

[1] Though many teams like to equate story points with time in some measure, that was never really the intent of story points. And, yes, technically, we're outside of the normal software development realm when it comes to story points, since we seldom, if ever, actually work from user stories.

[2] I'm not sure the Fibnoacci sequence was the chosen scale, but that's what stuck.

[3] Traditionally, I would argue that the overarching request should be treated as an Epic with individual stories representing the work necessary to deliver the feature. An Epic can span multiple Sprints, but Stories should not. That's not to say that Stories don't roll over from one Sprint to the next; they most certainly do, but the idea is that as a team matures and understands how to estimate accurately, the amount of rollover should go down over time.