What the Scrum Guide Doesn’t Say

Dan Ray Follow
May 13 · 8 min read ★

As “bibles of an industry” go, the Scrum Guide is downright *tidy*. At 19 pages (including cover page and table of contents) and fairly few big words, the text itself is quite accessible. But of course, anyone who’s dug into it far enough to really implement Scrum knows, those easy-to-read words open a rabbit hole that goes *deep*. As the Guide says, Scrum is lightweight, simple to learn, and difficult to master.

> *Scrum is not a process, technique, or definitive method. Rather, it is a framework within which you can employ various processes and techniques.*

My momma taught me that if you want to really understand what somebody thinks, you need to listen to what they say *and what they don’t say*. When it comes to processes and day-to-day operation of Scrum, there’s a LOT the Scrum Guide doesn’t say. Let’s look at a few of those.

Estimation

For a lot of teams, Scrum IS the mechanics of stories, points and velocities. But guess what? The Scrum Guide doesn't specify that estimations are to be done using that method, or any particular method!

The Guide only mandates that estimates happen. How they are done is left up to the team. The section "Product Backlog" contains the following references to estimation:

> *Product Backlog items have the attributes of a description, order,* ***estimate****, and value.*

> *Higher ordered Product Backlog items are usually clearer and more detailed than lower ordered ones. More precise* ***estimates*** *are made based on the greater clarity and increased detail; the lower the order, the less detail.*

> *The Development Team is responsible for all* ***estimates****. The Product Owner may influence the Development Team by helping it understand and select trade-offs, but the people who will perform the work make the final* ***estimate****.*

Get this: the word "velocity" shows up *exactly zero times* in the Scrum Guide. The Scrum Guide doesn't even *talk* about User Stories. Not a single "As a, I want, so that" in the whole Guide. The specific design of your backlog items, beyond the three data points mentioned above, is not something the Guide concerns itself with.

So what's the deal with the enshrinement of the Story Point as the be-all-end-all unit of estimation?

Well it comes from Extreme Programming, or XP. XP calls for the use of an organizational pattern called "Yesterday's Weather", in which abstract, relative measures of past performance are used as a guide for planning future work. Plenty of teams are having what appears to be plenty of success with Yesterday's Weather estimation, but it's certainly not the only way.

Scrum true-believers sometimes react in horror when they hear of a team estimating in terms of hours and days, but the Guide doesn't forbid that. If time-based estimation is working for a team, *that's not "Not Scrum"*. In fact, comparing estimated time versus actual time is a powerful (though easily abused) way of inspecting the process.

On the other hand, the #NoEstimates movement is growing, and is worth a google if you're unfamiliar. The idea is that if the stories are small enough, we get perfectly good planning done if we simply work with raw story count. The tiny ones and the little-bit-bigger ones all average out and we may as well just call them all "1-pointers". (To be clear, this is a deviation from the Scrum Guide, so make sure you fully understand that if you experiment with it. Lots of interesting debate is happening about #NoEstimates that I'm not going to rehash here.)

The point is that like with so many things, the Scrum Guide leaves us to empirically discover what method of estimation works best for us.

Self-Organization

For such a crucial Scrum concept, the Scrum Guide has shockingly little to say about self-organizing teams. The closest it comes to even defining the term is:

> *Self-organizing teams choose how best to accomplish their work, rather than being directed by others outside the team.*

…and:

> *[Development Teams] are self-organizing. No one (not even the Scrum Master) tells the Development Team how to turn Product Backlog into Increments of potentially releasable functionality.*

Regarding the closely-adjacent topic of accountability, the best we get is:

> *Individual Development Team members may have specialized skills and areas of focus, but accountability belongs to the Development Team as a whole.*

What this tight-lippedness implies is that there's a lot of room for a self-organizing team to work in, and a lot of ways they might *self-organize to self-organize*. We aren't given a prescriptive set of instructions about how to self-organize (hilarious though that would be). The Guide doesn't even paint us a picture of what self-organization looks like. Instead it leaves it for us to discover.

As I've worked with Scrum teams, I've tried to follow that example. It's true that showing a new team a model of what "self-organization" looks like might be an expedient shortcut, but I say that the process of fumbling toward it is where most of the growth opportunities for the team lie. The last thing I want to do is steal the team's "aha moments". Besides: expedience ain't everything.

Self-organization is an emergent phenomenon, an expression of the unique variations of people and personalities that make up the team. One team's self-organization is going to look different from another's. We want to embrace these differences, rather than trying to stamp out cookie-cutter versions of inherently messy things like human relationships and interactions.

People Leadership and Management

The Scrum Guide establishes a team structure that's not at all correlated to what most organizations think of as a reporting structure or organizational chart. In most organizations I've seen adopt Scrum, the Development Team members report to someone who's *not* the Scrum Master or Product Owner.

That means that Scrum isn't a management methodology. It's not motivational in nature (more on that later). It doesn't contain performance measures. Those things are all from a completely different context, called Management or Human Relations.

Most of the historical practices for performance assessment are based around individual goals and performance, which is a bit regrettable for Agile organizations. Agile is all about team, and not having at least a peek at team performance in the process of a employee's annual review is a shame.

I encourage organizations to start including those sorts of components in their people's performance evaluation process. Not to the exclusion of individual matters of course—these are individuals with individual career goals earning individual paychecks, after all. But their position on a team ought to be a consideration in reviewing the value they provide.

Now back to "Scrum isn't motivational". I hear the question from Product Owners sometimes, wondering how to motivate their teams.

The answer is: *You can't*. Motivation comes from inside, or else it's just behavior modification—some conditioned response to a carrot or a stick. Stop trying to motivate people, and start having the value of their work be transparent to them. Give them room to step up into their own maturity and personal bigness.

Pack mules might respond to "being motivated". Human professionals need to be empowered.

Stakeholder Management

This is a massive topic in traditional project management. It mostly comes down to two things: Managing expectations, and defending scope.

In a traditional organization, prioritization is mostly a HiPPO process: it's based on the Highest Paid Person's Opinion. "Stakeholder" is a shorthand for "someone further up the org chart than me". And that means they can jerk my project around if they don't like what they're seeing, or force us to work on some idea they have about what it should do. As a result, "managing" these "stakeholders" mostly means keeping their nose out of our business—a crucial task for keeping a traditional project from exploding.

Oh and then there's this: if they really don't like how things are going, they can fire me! How's *that* for psychological safety??

The Scrum Guide doesn't help much to define who counts as a Stakeholder, but it does talk a (very) little about what to do with them.

> *The Scrum Team and its stakeholders agree to be open about all the work and the challenges with performing the work.*

> *During the Sprint Review, the Scrum Team and stakeholders collaborate about what was done in the Sprint.*

> *At any point in time, the total work remaining to reach a goal can be summed. … This information is made transparent to all stakeholders.*

Compare this to the mountains of books about Stakeholder Management that come from traditional project management.

The key hint about the shift in orientation toward our Stakeholders in Agile comes from the section about the Sprint Review. Every time Stakeholders (or “all attendees”) are referred to regarding the Review, it’s in the context of collaboration, discussion, and mutual participation.

In Agile, we’re not dictated to by our Stakeholders. We’re co-owners of the value we’re providing. We leverage their unique perspective on the product, the organization, and the market to our mutual competitive advantage. Stakeholders aren’t scary anymore, they’re *useful*. Our job is to partner with them to deliver them value.

. . .

What "Not Saying” Means

It might be getting clear that it’s impossible to do *just* Scrum. There’s simply not enough detail in the Guide to implement a fully functional way of working. We *have* to supply our own methods for the things I listed above, and several others.

But we’re called upon to supply those things in a way that is informed by and harmonic with the approaches and philosophies embedded in the Guide. For instance, in an Empirical environment, how do you *know* that story-point estimation is what will work best for you? Well you don’t, unless you’ve tried it and inspected its effectiveness, AND tried a few alternatives to compare it with!

In my view, that’s why the Scrum Guide gives us things like Empiricism and the Scrum Values. Those “soft” features of the framework give us a place to stand as we put flesh on the bones of our day-to-day processes and operations.

So go boldly and discover for yourself a workflow that hangs inside the Scrum framework, consistent with and informed by the Scrum Values and the Pillars of Empiricism!

. . .