Functional and Non-Functional

MEASUREMENT AND REPORTING WITH SCRUM

By: Joe Schofield

Abstract: This article suggests an approach for the integration of functional and non-functional measurement for Scrum practitioners. Because Scrum is undeniably the most widely used of the agile frameworks,[1] clear articulation for measurement techniques with Scrum in general, and stories in particular, are a necessity for Function Point Analysis (FPA) advocates. Additionally, incorporating relevant measurement into organizational reporting for traditional cost, schedule, and scope is critical to enable and sustain agile cultural transformation.[2]

Why focus on Scrum and (User) Stories: Most organizations, as well as Function Point enthusiasts, struggle to transition from traditional "predictive"[3] development methodologies to adaptive frameworks. Many Key Performance Indicators (KPIs) and Objectives and Key Results (OKRs) are tied to classic "iron triangle" components of scope, cost, and schedule. Meanwhile, 66% of agile teams use Scrum, with another 15% using Scrum with either Kanban or eXtreme Programming (XP).[1] As an interesting side note, many teams using Scrum with Kanban, or ScrumBan, do so because they reject timeboxing which requires them to stop, inspect, and adapt to alleviate workflow impediments. Ironically, timeboxing with continuous improvement is quite often the very remedy to their inability to complete work within a sprint, their initial motive to abandon Scrum. Another often-cited reason is their perceived limitation to release "on-demand" with Scrum. Despite these curious criticisms, Scrum and its cited variants are used eight times more than all of the other agile frameworks combined. Organizations and teams utilizing Scrum are therefore the primary target audience for this paper. However, the common denominator for functional and non-functional measurement is a story, also used for instance, in eXtreme Programming as story cards.

Applying FPA to User Stories: Functional measurement quantifies value delivered to the business. The business is the recipient of a service or product created by the Scrum team as prioritized by the Product Owner, aka the "voice of the customer." In 2018 I proposed that stories be written at the elementary process level corresponding to CRUD (create, retrieve, update, delete) activity offering several advantages: [4]

1. Writing user stories in the language of the business aligns with FPA by keeping the focus on business needs rather than collecting technical implementation details which, are often mistakenly captured as part of the story rather than as tasks required to fulfill a story. The business need to "take an order" translates easily into tasks to "create an order" during sprint planning.

2. Perhaps the most valuable reason to decompose stories to the elementary process level is to answer the elusive question of "when do we stop breaking a story down?" The absence of a decomposition *boundary* of an elementary

process can result in excessive decomposition often to the task level, at which point the story loses its business focus. Unnecessary decomposition is not lean, nor is it in keeping with the 10th agile principle.[5]

3. Duplicate functionality becomes more visibile; re-use more intentional.

4. The story translates naturally as a transactional function for FPA.

5. The story is defined as "small enough" to fit within a sprint aiding teams that claim stories are "too big."

6. Acceptance criteria are narrowly defined promoting clarity and tighter "coupling" with story completion.

7. The data functions are typically specified as part of the story or included in the acceptance criteria for the story.

8. The acceptance criteria for the story provides a "container" for non-functional needs and considerations for IFPUG's SNAP.[6]

9. Teams can establish a taxonomy to retain and distinguish among the use of story points, function points, and even use case points for relevant audiences.[7]

Using the Acceptance Criteria in Stories for SNAP:[6]

Scrum stories capture the need of the business and when tasked during sprint planning, the work of the Scrum team. Briefly, not exhaustively, Scrum stories reside in the Product Backlog, are written in the language of the business, are owned and prioritized by the Product Owner, can be written in the Connextra style[8] (As a...I want . . . So that . . .), are limited to the effort and duration of a single iteration or sprint. Acceptance criteria is an essential, some would argue mandatory, attribute of a story. A story without acceptance criteria is not "ready" to be pulled into a sprint backlog. Once committed as part of the sprint backlog, the acceptance criteria remain unchanged. The demonstration by the Scrum team of the story and its acceptance criteria is the basis of the "acceptance" or "rejection" of a story near the end of an iteration during the sprint review. The Product Owner adjudicates the "acceptance" or "rejection" of the story on behalf of the business and its stakeholders. Acceptance criteria that are common to the work of the Scrum team may be consolidated into the "Definition of Done" (DoD), which all stories are expected to satisfy before being included in the sprint review.

Acceptance criteria combined with the DoD constitute the "conditions of acceptance" of the story; that is, the functional and non-functional verification (on behalf of the producer) and validation (on behalf of the consumer). Almost all of the non-functional elements of SNAP are well-suited for inclusion as acceptance criteria. The following table offers guidance for using the acceptance criteria associated with stories for SNAP determinations.

Table 1: Suggested SNAP elements incorporated into User Story Acceptance Criteria

SNAP Elements	Testable Acceptance Criteria?	*As an example . . .
1. Data Operations		
1.1 Data Entry Validations	Yes	A valid date; a valid address within a city
1.2 Logical and Mathematical Operations	Yes	Precipitation prediction based on historic probability
1.3 Data Formatting	Yes	Different date formats; credit card data tokenization
1.4 Internal Data Movements	Yes	Use a date from another "partition"
1.5 Delivering Added Value to Users . . .	Yes	Use the zip code to capture the city and state information
2. Interface Design		
2.1 User Interfaces	Yes	Enter a date via voice
2.2 Help Methods	Yes	Hover over a "?" to see how to enter or select a value
2.3 Multiple Input Methods	Yes	QR codes scan, links, or "taps"
2.4 Multiple Output Formats	Yes	Send a receipt via e-mail, a robocall, or text message using the same functionality
3. Technical Environment		
3.1 Multiple Platform	Yes	Web- and app-based solution
3.2 Database Technology	**No	
3.3 Batch Processes	Yes	A monthly compliance scan that does not report any user data
4. Architecture		
4.1 Component Based Software	**No	
4.2 Multiple Input / Output interfaces	**No	

*The "As an Example" column contains cells that are notional, an expression of an idea. They are not intended to redefine or alter SNAP definitions per the Software Non-functional Assessment Practices Manual (APM).

**The Scrum development team is responsible for the solution it delivers. Neither the Scrum Master nor the Product Owner determines who does the work of the team or how that work is completed; this includes but is not limited to database and component usage. Granted, the existence of architectural components and standards, platforms, and environments may be "inherited" by the team from the broader organization. "Scaling" choices may also constrain the technical solution that would otherwise reside with the Scrum team.

From *Measures* to *Metrics*[9]: Alternatives for Agile Project Reporting

As a quick reminder, the word *measurement* is used to describe the act of collecting *measures;* that is, values like quantity, weight, value, and size. *Measure* is also a verb that depicts the comparison of a value to a standard like inches, pounds, and hours. *Metrics* are the meaningful comparison of two *measures* to derive a value for comparison to similar paired sets of *measures.* As an example, Scrum teams may conduct *measurement* by collecting *measures* such as expected task hours, actual task hours, expected story value, and story points. Teams may derive *metrics* to improve future expected times by comparing expected to actual task hours. In addition, they might sum story points "accepted" per sprint to derive velocity.

Precautionary reminder: The expressed intent of this article is to more easily assimilate the continued use of FPA and SNAP in story-based agile environments. Attempts to use velocity, as an example, to compare teams or to forecast Scrum team completion dates by applying highly unstable and intentionally volatile product backlogs are harmful to self-organizing teams. This same misuse of measures impedes the desired trust between the Scrum team and its stakeholders. Often contrary to organizational desires, agile teams shun productivity measures within and among teams due to variation in the:

- target product or service,
- number and expertise of team members,
- physical and technical environments,
- profit-margin impacts,
- hidden investment in cross-functional development, and
- team role boundaries violated or honored by self-organizing teams.

Scaling frameworks that attempt to minimize some of the wariness and suspicion that accompany "productivity measures" can be met with skepticism, often well-deserved due to the organization's past practices. Providing the Development team with work estimates hinders self-organization and team accountability. Productivity measures and estimates are easily and often misused and manipulated10 with story points and projected completion dates. Leadership teams convene around conferences tables (virtual or real) and ponder how to get Team A to produce more like Team B instead of how to remove barriers to success that Team A confronts daily. Frequently, leadership inhibits the cultural change necessary for teams to succeed.[11]

Team Measurement

Measurement is conducted primarily for two reasons:

1) As clearly expressed in the 12th Agile Principle, teams measure to improve performance based on their continuous learning and ongoing improvement, the essence of heuristic thinking[12]. The Principle reads . . .

> ***At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.***

Observations made during retrospective events vary from teaming skills, to cross-functional growth, communication flows, tool knowledge, distraction management, self-imposed constraints, backlog refinement, and workflow. While measures can certainly be collected for each of these, other more readily, and often more practical measures are collected for the number of "accepted" stories, story points and expected and actual task time. From these measures, teams can if desired, understand variation in their estimates and actual task time, use the story points for a threshold in future sprint planning meetings, and minimize the effects of known causes for rejected stories. Each of these opportunities further illustrates the value of heuristic thinking.

Teams practicing Kanban are by definition more focused on work-in-progress (WIP) limits, as well as lead and cycle time optimization. While Kanban is not the focus of this article, it still promotes quantitative continuous improvement.

2) Measurement supports status reporting structures, traditionally for projects, programs, and portfolios. Less fortunately, measurement can be used as a defensive mechanism when the team is accused of being behind schedule, over budget, or outside the scope. But each of these "iron triangle" constraints is of decreasing value to agile organizations who find more meaningful metrics for understanding releases, value delivery, and prioritization.

Organizational Measurement

Project level measurements are almost always aggregated at some level in the organization. Mid-level management and senior leadership periodically scrutinize reports that consolidate key initiatives onto a dashboard. A classic summary of these project statuses include:

- projects in rows along a "y" axis,
- cost, schedule, and scope "columns" and
- red, yellow, green indicators in intersecting cells.

Reports like these provide decision-makers with a glimpse into what they believe to be the overall health of the work being undertaken, some financial confidence that dollars are not being expended without the completion of milestones (scope), and alignment with expected completion (schedule). Projects escape further surveillance when the cells are "green." Questions arise when the indicators are "yellow" and often attract unwanted attention when "red" or thought to be trending as such. "Fixing" the red indicators has for decades been addressed with change requests or exception reports that typically require an explanation for the deviation, some corrective action, and potentially a new forecast of completion, spending, or scope freezing. Organizations can develop onerous and invasive processes to raise leadership's confidence that tomorrow will bring better outcomes. Sadly, agile projects that foster changing requirements and priorities late in development, driven by business innovation, become the shamed victims of reporting systems that focus on milestones and outdated schedules. This culture of reporting and corrective action obscures agile transparency and stymies adaption by viewing scrum product increments through historically-tinted bifocals. Agile progress indicators are rejected or ignored. What are they?13

Agile's value-delivery emphasis is misrepresented by classic cost, schedule, and scope measures. Rather, that discussion needs to be recast toward *value-delivery, releases,* and *priorities,* respectively.

- *Value-delivery* is realized when "seed funding" encourages the team to create early value and then (funding) flows as value continues to be delivered. No or limited value delivery also serves to question the longevity of the work triggering the possible cancellation of work that has little chance of being completed. Contrast this scenario with projects that are funded for years that fail to deliver and where funds could have redirected toward opportunities with more promise.
- Early and frequent *releases* provide the business with early and frequent value since the business selects the capabilities to be delivered. Tracking releases and release velocity seems superior to setting and re-setting schedules.
- Scope has always been about *priorities;* "in-scope" is more urgent than "out-of-scope." But tracking priorities in Scrum reminds us that *priorities* are subject to change at the discretion of the product owner, potentially honed each sprint during *grooming,* also known as *refinement.*

Grooming is the Scrum change control process. Agile organizations do not need the burdensome change control processes imposed by traditional and well-intended project management offices for expectedly evolving and constantly innovated business needs.

Alternative Team Metrics

Since each sprint creates some value as determined by the Product Owner, the inclusion of a value with each story by the Product Owner enables the derivation of value for each sprint. The value-delivered for completed stories in each sprint is represented in Exhibit A. Tying stories to Function Points and SNAP Points provides an added value for FPA enthusiasts.

The value delivered for each release can be embodied in a cumulative value-delivery chart. Release when determined by the business, considering the cost of delay and the transaction cost of the release.[14] Again, the relationship of stories to releases and stories to Function Points facilitates FPA.

Exhibit A

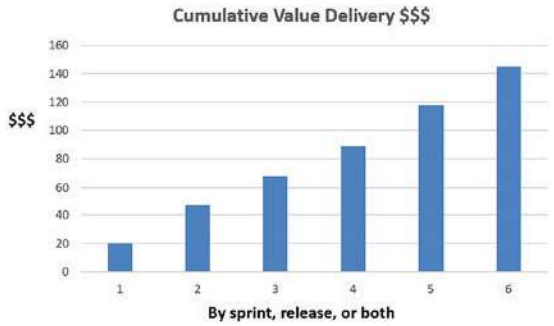
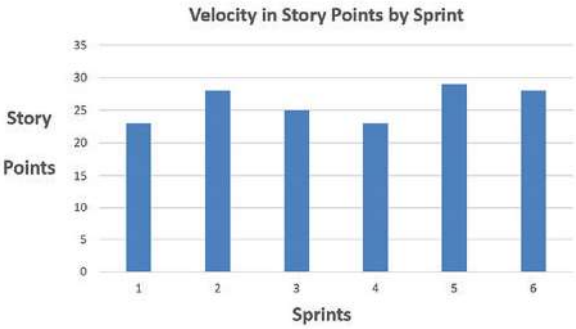


Exhibit B



Other insightful metrics might include defects per release, story point completion (velocity) to defects, and capacity (team available hours per sprint) to velocity. This last metric is an obvious contributor for explaining variation in velocity (see Exhibit B as an example), yet is often ignored. Absent this information, senior leadership may seek explanations for swings in velocity or even worse, attempt to compare velocities among teams in search of some best practice to be imposed on all teams. Understanding the nature of each team's relative values with story points and derived velocity is too often missed at the organization level. Establishing standards for velocity across all teams has other detrimental, even disastrous effects. The organization prioritizing a story point value over Scrum team understanding during its own grooming provides numerous opportunities for negative outcomes.

Conclusion

Agile product development occurs worldwide. Scrum is without a doubt the most prevalent agile approach employed. Similarly, IFPUG's Function Points and SNAP share a broad international audience. Traditional reporting impairs Scrum's value-delivery approach; however, as proposed, agile relevant success measurements and FPA can be employed bridging historic and contemporary measurement systems. Organizations that include senior leadership, management, and practicing teams as part of their agile transformation have a brighter glimmer of hope to overcome the cultural change that so often disrupts adoption.

Special thanks: Talmon Ben-Cnaan whose helpful insights and expertise enhanced the quality of the "examples" in Table 1.

REFERENCES:

[1]15th State of Agile Report; digital.ai; May, 2021; page 13 – Scrum used by 81 percent of survey respondents

[2]Ibid.; page 12 – the 2nd leading challenge to agile adoption is "organizational culture at odds with agile values"

[3]Aligning the PMO to Lead Agile Transformation; Schofield; Project Office Journal; IT Metrics & Productivity Institute; September, 2019

[4]Reflecting on Measurements in an Agile World; Schofield; MetricViews; International Function Points Users Group; Spring, 2018; pages 14 - 17

[5]https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/; Agile Principle # 10: Maximize the amount of work not done!

[6]https://en.wikipedia.org/wiki/SNAP_Points retrieved 10/4/2021; SNAP is the acronym for "Software Non-functional Assessment Process", a measurement of non-functional software size. The SNAP sizing method complements ISO/IEC 20926:2009, which defines a method for the sizing of functional user requirements. SNAP is a product of the International Function Point Users Group (IFPUG) and is sized using the "Software Non-functional Assessment Practices Manual" (APM) now in version 2.4. Also, ISO/IEC 32430:2021; The SNAP methodology has the IEEE standard IEEE2430-2019.

[7]Function Points, Use Case Points, Story Points: Observations from a Case Study; Schofield, et al; CrossTalk; May / June, 2013

[8]Rachel Davies; Connextra; circa 2000

[9]https://wikidiff.com/measurement/measure; retrieved 10/5/2021

[10]Inflate Gate: Mastering Overestimation for Agile Software Projects; Schofield; Computer Aid's Accelerated IT Success; (Featured Article); IT Metrics & Productivity Institute; August, 2015

[11]Aligning People and Culture for Agile Transformation; Schofield; 2020; See Vignettes 45, 48, 53 – 56 – Executive Action Team

[12]https://www.thoughtco.com/heuristics-psychology-4171769; retrieved 10/5/2021

[13]Countering 5 Barriers to Organizational Enterprise Success; Schofield; AgileConnection; July 17, 2019

[14]The Principles of Product Development Flow, Second Generation Lean Product Development; Reinertsen, D Celeritas Publishing; 2009; ISBN-10: 19354010

ABOUT THE AUTHOR

Joe Schofield is an Authorized Training Partner, a Scrum Certified Trainer and Agile Coach with SCRUMstudy™. He has more than 80 published books, papers, conference presentations—including contributions to the books: *The IFPUG Guide to IT and Software Measurement* (2012), *IT Measurement, Certified Function Point Specialist Exam Guide, The Economics of Software Quality,* and his recently released *Aligning People and Culture for Agile Transformation.* He holds nine agile-related certifications: SCT™, SCAC™, SSMC™, SSPOC™, SMC™, SDC™, SPOC™, SAMC™, and SAFe 5. Joe is also a Past President of the International Function Point Users Group, a Certified Software Quality Analyst, and was a Lean Six Sigma Black Belt. He retired from Sandia National Laboratories as a Distinguished Member of the Technical Staff. He served the last 10 years of that 31-year career as the "Chief Process Officer" of an organization of 400 software engineers which was awarded a SW-CMM® Level 3.