Measuring [Agile] Software Development Team Performance

by Kevin Loney

For further reading, see The Goal by Eliyahu Goldratt.

What measures do we use to determine if software development teams are improving, and how do we quantify that improvement? Through those measures, what behaviors are we rewarding and encouraging to be repeated? Or, consider the parallel challenges: How do we stop publishing metrics that misrepresent contributions from development teams? How do we avoid incentivizing behaviors that do not add value, or that create anti-patterns in the development process?

These questions don't get easier as a company grows and transforms. With the introduction of scrum teams — agile development teams that can include all functions required to design and implement a feature including a tech lead, developers, a business analyst, testers, and a coordinating scrum master — the performance evaluation process changes. Each person on a scrum team is an individual contributor but is also a member of a team, and it is the team that owns the ultimate deliverable. Rather than starting evaluations by looking at individual metrics, the focus in a team-based organization shifts to team-oriented outcomes and metrics. What are the most accurate metrics to guide judgments of their performance improvements and gaps?

Act 1. How many story points are allocated for this meeting?

It was a grey day in early January when the management team came together to address these exact questions. Their hope was that coming up with consistent metrics as the technology group went fully agile would coincide with the start of the year, and that employees' objectives for the new year would be written to align with these metrics, providing ongoing visibility to performance. The agile coach, the financial control lead, the HR representative, the project management office lead, and the development lead were joined in the conference room by a business analyst who would be a full participant in the session, challenging proposals as they drew on the white boards. And in the end, the analyst was the one who came up with the 'elevators' metaphor, with her perspective moving the discussion toward a real outcome.

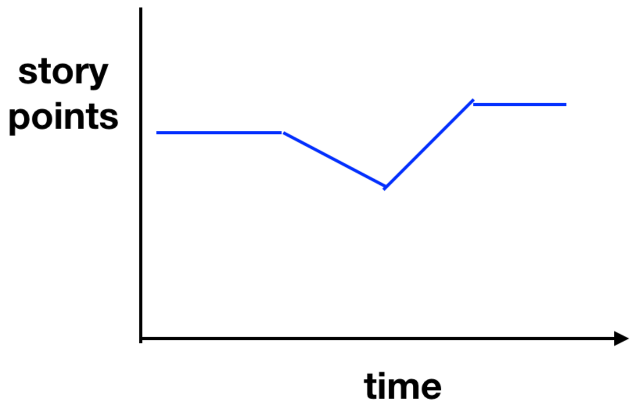
The project management office lead had started the discussion by revisiting the two perspectives on quality: from the internal functions view and from the external customer view. The internal view looks at how well something is made based on the recipe that was provided: how well does the finished product match the specifications? Were the features delivered as described in the high-level and low-level designs? Gaps against requirements are fairly easy for development teams to track and report on, so teams have historically captured those metrics. But when you are considering a cross-functional team, do those metrics apply to the whole team? After all, the tester did not develop the data model or the low-level design; is it right to hold the tester accountable for a gap between a low-level design and the deployed code?

From those questions, focus shifted to the external customer perspective: how well the finished product suits the customer. External quality measures (see the Malcolm Baldridge National Quality Award criteria, for example) focus on how well the delivered product meets a customer's need and provides value. In an ideal scenario, the development team's work is perfectly aligned with the features that will deliver the most business value the fastest. The customers don't care about the scrum teams' schedules; they care about their own calendars and their expectations of the features that will be available when the product is first released. The customers do not care about the number of lines of code or the programming language chosen. They care about the value that the product generates for them, and they want the underlying solution to be resilient, secure, scalable, recoverable, and highly available.

In organizations that had previously tracked metrics like lines of code, or the number of tests performed before code is released to production, how can the focus shift to something more relevant to the customer while the measure is still within the control of the development team?

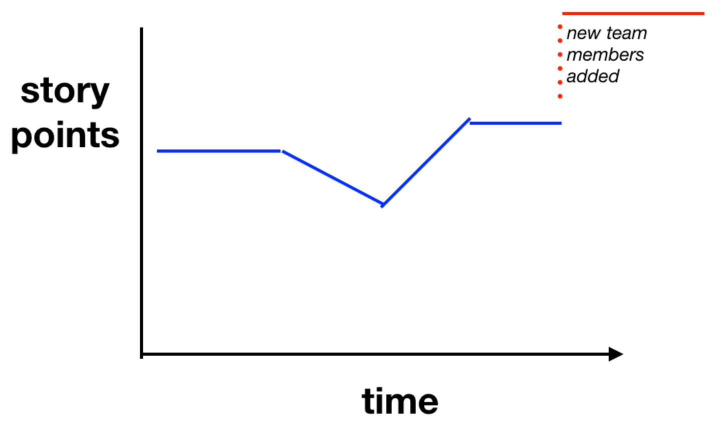
"How about using story points?" the development lead asked. Every feature was made up of components called epics, and epics were composed of stories, and each story was sized with a number of 'story points' when it was sized by the team. Stories were usually sized to be completed within a two-week sprint cycle. For each two-week sprint, you could tell how many story points' worth of work were being released to production by adding up the released work. He continued, "Just add up the number of story points for all the stories released to production in a release — called the velocity — and you can see if you are trending up or down."

The business analyst drew this on the white board. The agile coach winced.



"There are two problems with that," the coach replied. "What's the biggest factor that makes one team have a higher velocity than another team, as measured by story points? The headcount of the team, right? If you add more developers, you should be increasing your velocity. So this is an internal *leading* indicator of *potential* work activity, not an external *lagging* indicator of work value *actually delivered*. And it might seem like increasing the team potential should always deliver the final finished product to the customer faster, but we all know that's not a given. There's not always an economy of scale. Customers don't get the full value of the delivered product until all the stories in the related epics are delivered. If there are dependencies, or constraints, or if the additional workers don't work on the critical stories, or if you have the wrong types of workers, then there is no change to the delivery time for the feature even though more story points are completed. It may even slow down.

"I know we're not supposed to say that story points correlate to man-hours," she continued. "So let's just stay there is an amazing coincidental change in values so that both go up and down together." With that, she took the marker and extended the velocity line, adding a significant step increase to the velocity at a point marked "new team members added."



The agile coach handed the marker back to the business analyst. "And hiring, and the funding for the hiring, and everything that goes with that, is in the hands of management, not the developers and testers. You can't hold developers and testers primarily accountable for staffing. If you're judging based on story points, they are constrained by a hiring process outside their control. You can't judge —or reward! — the team on its story point velocity."

"But there is at least a little internal value in knowing your velocity, or at least the derivative of the velocity," replied the development lead. "In our tracking systems if a story is delayed — by environmental issues, or personnel, or a weather event, or a design issue, or a testing issue — then you don't get credit yet and your story point velocity this release will drop; you get the points next release when the code ships. I want to know that; I want to see if that drop is localized to one team, or if there are multiple teams that got hit by the same issue. As a manager, I need to know who is struggling, which teams are having difficulty working well through their challenges? Are there group-wide issues? Are they splitting stories? Are they underestimating the duration of work, repeatedly? Are they pushing stories to the next sprint repeatedly? Are new team members helping or slowing things down? Something is wrong, so give me a chance to fix it. I realize the customers don't need to see the team member headcount change — it's like a restaurant telling customers when they hire more dishwashers — I just want my order to come out on time, the way I ordered it."

Act 2. Just because it's easy to count doesn't mean it's good math.

The business analyst turned to the table and said, "Let's consider those new team members. Let's assume they are fully able to contribute on Day One. How do we know if they are helping?"

The financial control lead pulled out a spreadsheet. “We have that already,” she replied. “We know from timesheets who is working on a project and who is billing overhead; how long it takes a new person to get through the corporate compliance training modules the first week. We can see how utilization drops during peak holiday times, for instance.”

“Utilization?”

“The percentage of their time that a person is billing to a project. Out of the hours they have available, what time is spent billing, and what time is spent on overhead tasks or personal time? That’s their utilization percentage.”

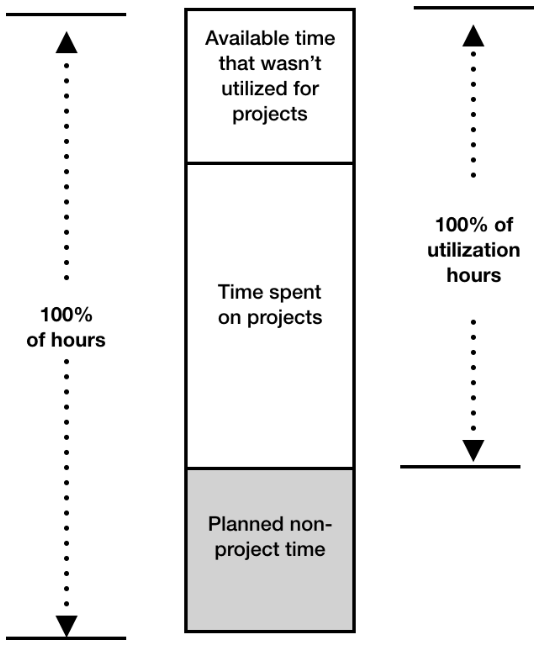
The HR rep shook his head. “People are allowed to take vacations,” he replied. “You can’t hold that against them. They actually perform better when they come back. And we want them to improve their skills, that’s why they are supposed to have training days set aside. It’s a strategic benefit for the company and the people. If anything we should consider funding learning-focused sabbaticals.”

“But it’s easy to track from the timesheets,” the financial controls lead argued. “We have the data, and we can see who is not actively utilized —”

“—which doesn’t tell you anything about why they are allocated the way they are,” the development lead finished. “They could be the subject matter expert for a legacy system and they keep geting pulled into incident triage. Or…or they could know multiple areas well and they are asked to analyze the viability of new strategic initiatives that aren’t yet projects. Or who knows — they could be tasked with coming up with an SDLC and the deliverables and measures used to demonstrate control effectiveness. They could get pulled into meetings like this. All of which would be unbillable work if you’re judging things solely based on corporate project schedules.”

The business analyst joined in: “What if we remove the planned outages, the vacation time, learning days, things like that. Can we then look at the number of hours remaining for each person, and see how much they were utilized?”

As she asked this, the analyst drew a stacked bar chart on the white board:


Available time that wasn’t utilized for projects
Time spent on projects
Planned non-project time
100% of hours
100% of utilization hours

“That seems reasonable,” said the finance controls lead. “You can see who is used during the time they had that was fully available.”

“Which still tells you nothing about the individual performance,” complained the development lead. “It’s just one number divided by another number. It still doesn’t address any of the issues already called out — you’re penalizing people for being the subject matter experts who resolve incidents, for one thing.”

“Does it tell us something about the whole team?” the project lead asked. “Is the whole team working on non-billable work at the same time? Isn’t anybody doing anything? How are we even forecasting end dates for projects?”

“They’re teams,” the agile coach answered. “Like relay teams. Each member has a different function. You don’t look at the individual performer, you look at the team performance.”

Act 3. Let me run this by you.

“The typical example to use here is relay runners,” the coach continued. “Say it’s a four-person relay. What is their goal? Is it for every runner to be constantly running, constantly utilized in this meeting’s vernacular? No. It’s for the final team member to deliver the baton across the finish line as quickly as possible. <u>It’s that final delivery that is the goal</u>. We all get that, right? That’s what the measures should focus on. Along the way, there will be times when all of the runners will have down time, either waiting for their turn to run or having completed their part of the relay. And that is okay. Their activity level at any one instant is *not* the goal. Their time spent running with the baton is part of their work effort, but we don’t hold against them the time they spent while the others were running.

“Let’s say the first runner completes a lap of the track and hands the baton to the second runner. The first runner doesn’t then go and get another baton and run another lap. We all know that. But mathematically, the first runner is no longer ‘utilized’ for this baton’s delivery. She’s finished her part of the work of the team. For the first leg of the relay she was 100% utilized; for the next three legs she will be 0% utilized. *And that is the plan*. We all know this. Because we all know <u>the goal</u> is to get the baton delivered across the finish line as fast as possible without dropping it, and to do that she has to hand it off to others.

“And what about the fourth, final runner? For the first three legs of the relay she is waiting. Should she spend that time running up and down a mountain so that she will be constantly ‘utilized’? Of course not — if she was constantly running up a mountain while waiting, that would negatively impact her ability to run at peak performance when it is her job to take the baton and complete the relay for her team. She should be monitoring the performance of the first three runners, coaching them when possible, and preparing herself to run the anchor leg. She should know where they are on the track, what’s coming her way, and how fast she’s going to need to go to finish at their target time. She should be aware of the track conditions — are there slippery spots? She should know about the environmental factors that could impact her — are there other teams running at the same time that could slow her down by jostling her? Have her teammates been struggling with certain parts? She needs to plan all that out while she is waiting her turn so when she actually does her part of the collective work — carrying the baton — her preparation, which was actually work but nobody was watching, sets her up to complete that work as effectively as possible for the team. So if we focus only on utilization —”

“— then it’s like having the elevators malfunction,” nodded the business analyst.

Act 4. The Ups and Downs of Elevators.

The leads seated at the table looked at the analyst quizzically. “Elevators?” asked the HR rep.

“Sure,” she replied. “We’re here on the 23rd floor, right? None of us walked up here. Everybody took an elevator. I got here early and there was nobody else in the lobby, and when I hit the call button for an elevator I could hear multiple elevators start heading down to me. The first one reached me and opened, and I could hear the others stop.”

“That’s how these elevators work,” the program lead smiled.

“Exactly,” the analyst replied. “That elevator lobby has eight elevators. When I arrived, none of them was moving. Then I hit a button and several started moving until one arrived, and that one moved until it took me to the right floor.”

“What’s the point?”

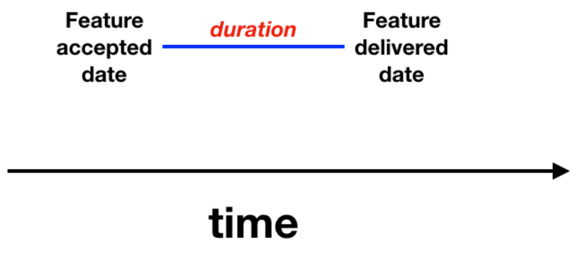
“The point is, if the target for the elevators was a utilization percentage of 100%, what would they do? They’d be constantly in motion. Always going up or down, stopping at floors randomly and opening and closing their doors. Maybe they hit your floor, maybe they don’t. You may wait a while for one and then ride it for a while until you get where you want to go. It would be a nightmare from a customer experience point of view, right?”

“Sure.”

“But from a *utilization* perspective, they’d be 100% utilized. Always on the move. And they’d have people in them for longer, so their loaded utilization would be increased too.”

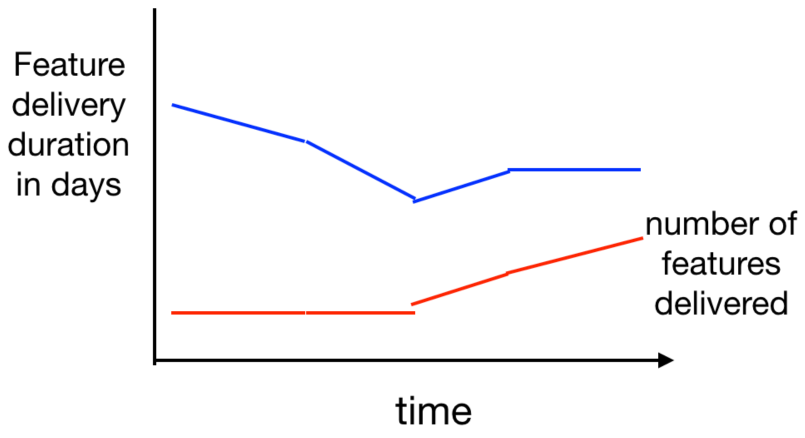
“But that’s not how you judge an elevator,” replied the project lead. “You judge it by how quickly it goes from the time you press the button to the time you arrive at the floor you want.”

“Precisely,” the analyst replied. “And that’s how our customers judge things.” She drew on the white board:



“There’s a date on which the team accepts a feature for development. And there’s a date on which the feature is delivered to production. From the customer’s point of view — the product owner’s point of view — those are the points that are critical and relevant. The day I buy something and the day it’s delivered. Everything else is just noise and status reports I don’t read. Now to see if a team is improving, track the changes in *that duration* over time, along with the number of features delivered. Is that duration getting smaller over time?

“So let’s chart this out. Try this. The blue line at the top is the duration for the features that were delivered that day. So that says how long did the customer wait for the value-generating deliverable that was delivered that day. And we want that line to continually go down, just as you want your total time for an elevator ride to go down.



“And the red line at the bottom is the number of features delivered in the release. This would provide context for the delivery duration data; did we deliver more in less time, or was the average delivery time impacted because we had more complex testing to perform when we had more features released at the same time?”

The development lead nodded. “Right. That’s a team effort. And it shows how the team works together to deliver value. I think it also helps the team members if we communicate it this way; it relates their work to delivering value for the customer pretty directly. That’s important for morale, for alignment, for bridging the gap between tech and the business.

“And these are the positives,” he continued. “In terms of negative impact for customers, we already track the customer impact of defects by month and by release. We could pull the number of defects per release and show that trend on the same chart. We could correlate the quality of the feature to the delivery duration of that feature and the number of features released at the same time. We’d see how much our defect rate is being impacted by our number of features concurrently delivered. Is our WIP limit too high?”

The HR rep added, “And you could see if teams were improving their feature delivery times, or if they were slowing their delivery in order to game the metrics elsewhere. We should talk through the scenarios so we understand how to evaluate these — what it means when duration values go up, down, or stay constant while the other metrics, the number of features delivered and the number of production defects, also go up and down.”

Act 5. Please stand back from the closing doors.

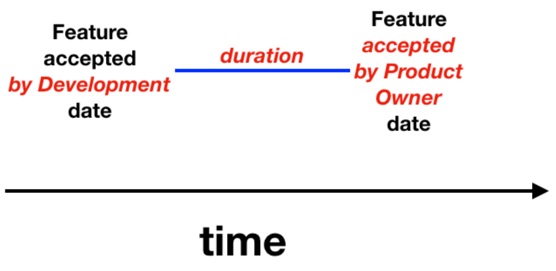
The leads sat back in their chairs and looked at the board in silence for a moment.

The financial controls lead finally shook her head. “We’re missing something,” she said. “This whole process starts with the customer and the technology group agreeing on the feature and the technology group accepting it for delivery —”

“Right,” agreed the agile coach.

“So it can’t end without the customer,” she concluded. “We can’t say that we’ve delivered something, and therefore that is the end of the timeline for that feature. That feature delivery timeline doesn’t go from ‘Feature accepted date’ to ‘Feature delivered date’. It goes from the ‘Feature accepted by Development date’ to the ‘Completed feature accepted by Business date’. Because you could develop something that misinterprets a requirement, or doesn’t address all the needs the customer has Day One. Like, you can’t deliver a car to someone and then come back later and add a windshield to it because that’s an add-on Day Two story.”

“In the agile world, we’d use the term ‘Product Owner’ rather than Business,” the analyst said as she erased portions of the earlier timeline and wrote:


Feature accepted *by Development* date
duration
Feature *accepted by Product Owner* date
time

“Now the duration reflects the customer’s total wait time,” added the project management lead. “And the team, the whole team, needs to go into development fully understanding the criteria that the business will use to accept the feature. It’s not just the time to follow a recipe. The test has to be complete. The requirements have to be correct. The code design should be solid so future changes will be simple and fast.”

“But not all product owners are the same,” warned the agile coach. “Some are good at defining what they want, some aren’t available all the time, some are real sticklers for performance—”

“— then the delivery teams need to spend more time with their product owners,” replied the testing lead. “Using an acceptance-based metric does not give the delivery teams the right to act like victims. It gives them the *privilege* of being partners with their product owners.”

The HR rep nodded at the board. “So you have the time people wait for the elevator to start, plus the ride time, plus the delays while it stops at the floors they don’t want until finally they get where they want.” Everybody nodded.

“Let’s sketch out the appraisal approaches over lunch,” the analyst proposed, checking her phone. “We have the room. And there’s a delivery guy here, but he’s stuck down in the elevator lobby.”

— Kevin Loney, January 2020. For further reading, see The Goal by Eliyahu Goldratt.

Any resemblance to any real people, companies, roles, buildings, elevators, or performance metrics is unintentional and purely coincidental.