An Appropriate Use of Metrics

Management love their metrics. The thinking goes something like this, "We need a number to measure how we're doing. Numbers focus people and help us measure success." Whilst well intentioned, management by numbers unintuitively leads to problematic behavior and ultimately detracts from broader project and organizational goals. Metrics inherently aren't a bad thing; just often, inappropriately used. This essay demonstrates many of the issues caused by management's traditional use of metrics and offers an alternative to address these dysfunctions.

19 February 2013

Patrick Kua

Patrick Kua leads development teams drawing upon agile methods to deliver valuable software for customers. He is author of The Retrospective Handbook: A guide for agile teams

Find **similar articles** to this by looking at these tags: metrics · productivity · project planning · technical leadership

Contents

Sidebars

> *Tell me how you measure me and I will tell you how I behave.*
>
> *-- Eliyahu Goldratt*

What's wrong with how we use metrics?

Organizations looking at metrics from a management by numbers perspective follow a process that looks like this:

1. Management come up with a goal and work out a measure
2. Management establish a target over a large period (3-6 months up to a year) for the people doing the work
3. Management communicate only the target (in terms of the agreed metric)
4. People doing the work do everything in their power to meet the target number

This process encourages overloading a metric with the following purposes:

- Metrics as a target – Numerical metrics make it particularly easy for people to use it as the only means for communicating a goal. It's often much easier to tell people a scale and a number than explaining a much more complicated goal. The target is frequently an arbitrary number and some organizations even spend excessive amounts of time determining what that number should be.
- Metrics as a measure of performance – With an established number in place instead of a well-articulated goal, it is now easy for managers to use that same measure to track how quickly people doing the work move towards the goal. Many organizations link these numbers to individual performance targets.
- Metrics as best practice – Using metrics both as a target and a measure of performance results in an unintended side effect – an implication this metric is the best method of working towards the goal. When an independent party measures someone else using a numerical target, it applies more pressure on the person doing the work to simply meet an established number. Since they are only measured on performance to this metric, they do all that they can to achieve that particular metric. It implies no other method is best at achieving the end goal.

Overloading a single metric with multiple purposes causes many problems, particularly when dealing with knowledge work such as software. Metrics are simplifications of much more complex attributes. The cost of simplifying complexity comes at the cost of losing sight of the real end goal, and ends in a suboptimal result.

Let's look at an example:

> A test manger, let's call her Mary, holds weekly meeting with the development lead, Dan. "Where are we at with our bug counts?" she asked at their most recent one. Dan answered, "We cleared our three priority one bugs, fixed four priority two bugs and cleared out a record twelve priority three bugs. A pretty good week right?"
>
> Looking at the development lead, slightly shaking her head, Mary responded, "Unfortunately our customer reported five priority one bugs, six priority two bugs and fifteen priority three bugs. You'll need to work harder next week." Exasperated and feeling overwhelmed at missing his target, Dan left the meeting thinking about asking his team to work yet another weekend.

In this very simple story, the chosen metric meets one benefit of making the meeting move very quickly. Both people quickly understand progress after Dan reports his results and when Mary responds. Unfortunately the implied goal of delivering useful software is missed, and Dan leaves the meeting with a solution more likely to cause further software issues and drag in software quality.

The way that Mary states her objective puts pressure on Dan to reduce the number of bugs. It seems like an admirable goal. While reducing the number of bugs is a good goal, it also leads to a very reactive solution. Dan leaves the meeting thinking how much harder to work. The question posed by Mary fails to neglect the broader goal, and she fails to ask the crucial question that guides Dan and his team towards fixing the underlying reason the bug exists. Without resolving this root cause, Dan and his team are destined to fix bugs for life.

Dan is experiencing single loop learning[1]. Single loop learning is the repeated attempt at the same problem, with no variation of method and without ever questioning the goal. If

Dan ever hopes to break out of this vicious bug cycle, he needs to do something differently. The inappropriate use of software leads Dan away from the end goal of delivering useful software and improving overall software quality. Einstein's definition of insanity seems to fit well here: "doing the same thing over and over again and expecting different results."

Be careful what you measure

Organizations love metrics because it makes setting targets easier, and discourages people from questioning the goal behind the target. This leads managers into a false sense of organizational efficiency. Strong incentives tied to strong metrics force people to concentrate on just one part of the work, neglecting other contributing factors that might make a goal more successful. Organizations must be wary of this actively destructive focus that leads people to neglect other important factors.

Even agile techniques do not protect teams from the undesirable behaviors driven by measuring and tracking the wrong number. For example, agile teams often use story cards[2] for development work. Teams often visualize these small increments of work on board as it moves through their organization's software lifecycle. A typical process might look like this with the ideal flow of stories moving from left to right:

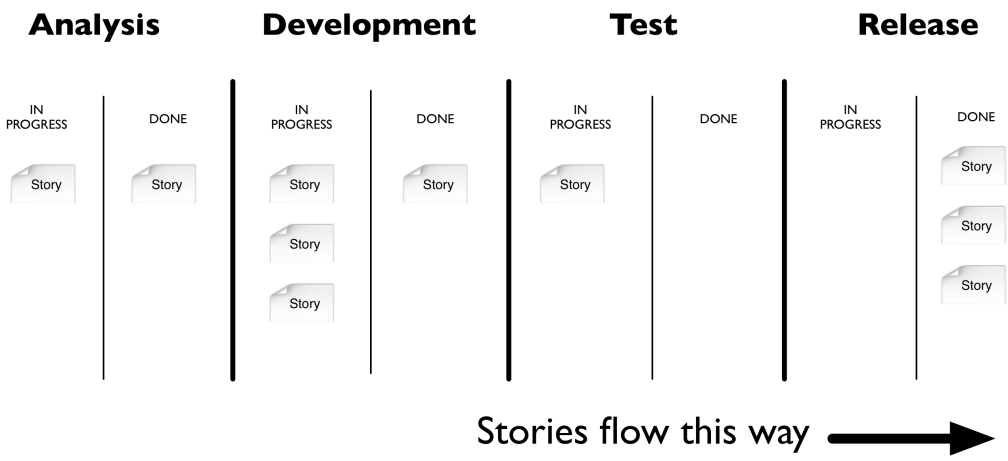

Figure 1: Example of a storywall

Management and product management often ask the question, "How soon will that feature be complete?" Teams often choose to interpret this as when coding finishes, succumbing to the idea that testing and the path to production are trivial and inconsequential parts of software process. Project management reinforces this perception by asking the question, "How many stories did we finish coding this week?" instead of the better question, "How many stories are we happy to release to end users?" or better yet, "How many stories did we release to end users?" An even better question is, "How much value have our users found from our recent releases?"

Teams want to do the right thing and these questions and metrics consequently drive developers to focus on getting stories*Development Complete*. Let's look at the consequences of overtly focusing on this sub optimal goal alone:

passage 1

> Malcolm, a marketing representative always takes a keen interest in what developers built for him that he dropped by the team as often as he could. He often talked to Dan, the developer, asking

when his features would be complete. Dan, not wanting to disappoint Malcolm worked hard to focus on finishing whatever Malcolm asked, knowing he wouldn't be far off from returning to ask on progress. He'd often think to himself, "This feature must be really important." Tim, the team's newest tester often needed to approach a developer, like Dan, to understand how to trigger the newly developed features.

Tim approaches Dan one day, "Hi Dan! I really need your help to understand how to test this feature you completed last week." Dan, under pressure to deliver snaps, "Can't you do anything by yourself? I need to get this feature complete so Malcolm gets off my back." Shocked at Dan's response, Tim returns to his desk, and waits. He thinks to himself, "I can't get anything done until Dan helps me out."

Each week this happens, and over time, the stack of stories waiting to be tested grows and grows. Eventually Malcolm calls a meeting with the team concerned he's yet to see that feature he asked for two months ago in production. Surprised, Dan says he completed it over a month ago. Tim bashfully responds, "I couldn't test that story because I needed some help from Dan and he's been so busy with other work. I didn't want to interrupt him."

What can we learn from this story? Firstly, what matters to Malcolm is that the flow of work is getting done. Even though Malcolm asks when something will be completed, what he really wants is to be able to use it in production. We know that Tim didn't have knowledge necessary to complete and his work and the pressures on Dan to complete more work prevented Tim from acquiring any more knowledge. The end result was a vicious cycle of work building up in testing, never getting released and with Malcolm puzzled why he hadn't received the feature he'd asked for. This is why methods like Kanban Software Development encourage *Explicit Work in Progress* limits. These limits force people to help out other when bottlenecks appear. These WIP limits work to overcome the undesirable behaviors that emerge when people are measured by the wrong metric of their individual productivity instead of overall value delivered. The book, Lean Software Development, stresses the importance of measuring the end to end result instead of simply a small part of the process, calling out a principle they call 'Optmize the Whole'. Optimizing the whole means ensuring the metrics in use do not drive sub optimal behavior towards the real goal of delivering useful software.

Guidelines for a more appropriate use of metrics

Given the undesirable behaviors that emerge due to the inappropriate use of metrics, does this mean there is no place for them? Of course there is a place for metrics. What is needed is a different method. Use the following guidelines to lead you to a more appropriate use of metrics:

1. Explicitly link metrics to goals
2. Favor tracking trends over absolute numbers
3. Use shorter tracking periods
4. Change metrics when they stop driving change

We'll use the following sections to explore what these mean.

Explicitly link metrics to goals

In the traditional style, management decides what the best measure for a particular goal is. Management then set a target in terms of that measure. Management then articulate only this target to people doing the work, in its, often, numerical representation. The lines between the measure chosen to monitor progress towards the goal and the actual goal itself blur. Over time, the reason behind the measure is lost and people focus on meeting the target even if that metric is no longer relevant. A more appropriate use of metrics is to ensure that the chosen measure for progress, the metric, is teased out, yet related to its purpose, the goal.

For example, in a software development context, you might see metrics defined like this:

> Methods must be less than 15 lines. You must not have more than 4 parameters to a method. Method cyclomatic complexity must not exceed 20.

With an appropriate use of metrics, every single measure should clearly be linked to its original purpose. The current mechanism for tracking and monitoring must be decoupled from its goal and that goal made explicit to help people better understand the metric's intent. A metric in a richer context for its existence guides people in making more appropriate, pragmatic and ultimately useful decisions towards the goal. Without its purpose, the effort expended means people find ways to creatively game their system, ultimately detracting from the real goal. Here's what that looks like:

> We would like our code to be less complex and easier to change. Therefore we should aim to write short methods (less than 15 lines) with a low cyclomatic complexity (less than 20 is good). We should also aim to have a small handful of parameters (up to four) so that methods remain as focused as possible.

Passage 2

Explicit linking the metrics to the goal allow people to better challenge their relevance, to find other ways of satisfying the need, and to help people understand the intent behind the numbers. Without this articulated purpose, people may find ways, unintentionally working against the implicit goal. For example, a number of techniques might help reduce a method length, but increase overall complexity by being harder to read if not applied with the correct intent.

The nature of software development means most work is knowledge work, and is therefore hard to observe. It is easy to monitor activity (how much time they sit at their computer) yet it is hard to observe the value they produce (useful software that meets a real need). The further that people move away from the code, the harder it is for them to appreciate the complexities involved. This implies that it is very difficult, if not impossible for people furthest away from the work to really know the best measure to monitor for progress towards the goal.

A shift towards a more appropriate use of metrics means management cannot come up with measures in isolation. They must no longer delude themselves into thinking they know the best method for monitoring progress and stop enforcing a measure that may or may not be the most relevant to the goal. Instead management is responsible for ensuring

the end goal is always kept in sight, working with the people with the most knowledge of the system to come up with measures that make the most sense to monitor for progress.

Favor tracking trends over absolute numbers

Management find metrics too hard to resist because it distils down organizational complexity into something everyone can understand, a number. It's easy to see one number is bigger or smaller than another, or how distant one number is from another. It's much harder to see if that number is still relevant. This traditional approach to management likes using these metrics because it makes it easy to communicate when a target is met. "Just reach this number and we'll be fine".

When you turn a qualitative and highly interpretive issue (think of productivity, quality, and usability) into a number, any figure is relative and arbitrary. There may be significant difference between code coverage of 5% and 95%, but is there really a significant difference between 94% and 95%? Choosing 95% as a target helps people understand when to stop, but if it requires an order of magnitude of effort getting that last 1%, is it really worth it? This is only something that people must work out subjectively in their own organizational context.

Looking at trends provides more interesting information than whether or not a target is met. Working out if a goal is met is easy. The difficult work, and one that management must work with people with the skills to complete is looking at trends to see if they are moving in the desired direction and a fast enough rate. Trends provide leading indicators into the performance that emerges from organizational complexity. It is clearly pointless focusing on the gap in a number when a trend moves further and further away from a desired state.

Focusing on trends is important because it provides feedback based on real data on any change implemented and creates more options for organizations to react. For instance, if the team is trending away from a desired state, they can ask themselves what is causing them to move away from their goal and what can they do about it. It pre-empts action much earlier than simply doing as much as they can before working out a number. If a team find themselves trending towards a desired state, they can ask themselves what is helping them move towards their goal and what else can be done to accelerate that rate. Measuring teams encourages people to experiment much more. Tweak one thing and observe its effect on the trend, monitoring where you are with the desired state and knowing when to stop.

Arbitrary absolute numbers also create helplessness, especially when progress towards a goal is slow and dependencies on other departments or corporate policies outside of a group's control prevent more progress. Trends help focus people's efforts on making movement in the right direction rather than being paralyzed between a gap that looks impossible to resolve.

A more appropriate use of metrics requires more management involvement in reporting and recording movements in trends because the ecosystem that surrounds a team is management's responsibility. This ecosystem includes the organization's policies, the way work is scheduled or planned and the way that teams and people are organized. This ecosystem often has much more influence on the trend that the efforts expended by individuals. Management should be interested in trends to observe the effects of changes to this ecosystem.

An appropriate use of metrics finds trends much more useful than absolute numbers. Arbitrary targets don't really have much meaning without the right trend and better questions emerge when thinking about what affects a trend

Metrics as a ratchet

Passage 3

ThoughtWorks are often asked to rescue software projects where a single change takes too long. What is viewed as a small change often takes upwards of a month to make. These type of projects share a very common trait - code quality treated as an afterthought and a significant amount of technical debt already incurred.

The codebase of a typical rescue project is rife with large code smells. They may be large in frequency, with a single code smell smeared across in many areas like many parallel inheritance hierarchies. Other code smells large in magnitude, like an excessively large method. In the worst case, a code smell is large in both frequency and magnitude. Tackling any of these appears impossible at first because the effort to fix one is overwhelmingly. Consistently addressing a single code smell is difficult because it requires putting an immediate halt on delivering incremental business value.

Reversing the trend of poor quality is the only way forward, made possible through ratcheting, a term my colleague Chris Stevenson wrote about in his blog article, "Fixing Broken Windows with Ratcheting."

Ratcheting involves adding a code analysis tool to a continuous integration build that fails when a certain metric exceeds a certain value. Teams start off with this very high. Adding it to the continuous integration build from trending further in the wrong direction. The team incrementally works at addressing the chosen code smell one small step at a time at the same time as delivering other functionality and business value. On each small improvement, the team revises the current value downwards.

What seemed like an insurmountable problem is taken apart one small piece at a time. With a ratchet put in place to prevent backwards movement, each small improvement moves the trend in the right direction.

and what else can be done to affect the trend, rather than pointing about what the gap is between an arbitrary number and reality.

Use shorter tracking periods

Many organizations use metrics to set targets for very long periods, typically 3-6 months, even as long as a year or beyond. Managers establish this target, with the responsibility lying with the people doing the work to do whatever they can to meet that target. Management revisits this target at the end of the period to *evaluate* the people doing the work. In this system, the relationship between management and the workforce is, at best, described as confrontational. The workforce, doing their best endeavors, do whatever they can to meet the goal, with an implicit idea that management do not have any responsibility.

A consequence of revisiting metrics after long periods is that the failure to meet management's arbitrary target becomes more and more unacceptable. I've heard managers say things like, "You had a whole year to meet your target and you missed it." The risk and cost of failure increases the longer the tracking period is.

Agile methods prefer shorter periods for review because any performance gap is less costly. Failing to make enough progress in a week is much less significant than failing to make enough progress over a whole year. Reviewing progress after each week generates many more options than reviewing progress after a year, simply because there are more opportunities to react and change. After a short period such as a week, you also have much more data about what actually happened instead of what was planned, and this should be used to influence the outcome by using it to drive change.

Organizations benefit from using shorter tracking periods as it creates more opportunities for re-planning that allows maximum value.

> I worked with a team that released software into production every two weeks. The business liked regular releases because they could use the software almost immediately. On using the software deployed after the latest release, the business discovered they had enough features they could do almost everything they needed for a new marketing initiative. It was only a fraction of what they originally asked for.
>
> Instead of the development team writing features that would probably never be used, the business picked a small subset of the leftover stories and started work on the next initiative.

An appropriate use of metrics tracks progress in smaller cycles because it gives much more information about where a project may end up further in the future. Tracking smaller periods helps identify trends and the pause gives organizations a more informed position to influence the environment and the rate/direction of a trend.

Tracking smaller periods also enables more collaboration because it provides more opportunity for management to be involved. Rather than simply *evaluating* people at the end of a larger period, tracking smaller periods provides more data about what is actually happening that influences the trends.

Change metrics when they stop driving change

If organizations reached goals easily, they would never need metrics. The organization could shift direction and they would reach their goal immediately. Unfortunately this doesn't happen in reality which is why measures exist. Achieving a goal often takes much longer. The first guideline to an appropriate use of metrics separates the real goal from the measure selected to monitor progress towards that goal. The real goal must always be made explicit.

Guideline #2 and #3, monitoring trends and doing so over shorter periods is about helping organizations realize their goal faster. It isn't achieved through the single-loop learning described earlier in the chapter. Organizations require is the double-loop learning Argyris writes about. An appropriate use of metrics drives people to question the goal and, based on collecting real data, implementing change to get there.

Here's what double loop learning looks like:

> Frustrated by fixing bugs every week Dan the developer considers why he is constantly fixing bugs. Over the last three weeks, Malcolm reports many issues about things not working as he expected. He steps back to think about what is really going on, less concerned about the bug count he is always asked about and more about why he has them to begin with.
>
> When Dan picks up a story, he often has lots questions for Malcolm about how it should work. Dan knows Malcolm has his other marketing activities keeping him busy and understands Malcolm cannot sit with him to answer his questions. Dan is under enormous pressure to deliver something, so he makes several assumptions to ensure he can deliver something instead of nothing.
>
> Looking at the bugs, Dan realizes that many of the bugs reported are based on those small assumptions he keeps making. The pressures to deliver something mean that Dan never builds the right thing the first time around.
>
> When Dan explains this to Malcolm, they agree to sit down at the start of each new story to make sure all of Dan's questions are answered before he starts coding. They try this the next week and the overall number of bugs reported that week decreases.

Double loop learning requires more data about what is actually going on. Shorter periods create more data points, making it easier to see any trends. The trends offer insight into the current performance of the system and should be used to trigger thinking and problem solving about the deeper underlying forces at play in the system, not simply for tracking performance measurement. Implementing real change helps accelerate organizations towards their current goal.

Changing the system that people work in often has a much greater impact than focusing on the individual's efforts to work harder or faster. In our story, Dan could have spent more time each week trying to fix bugs, but by adjusting the flow of information and the working relationship between Malcolm and Dan, they changed the system to be much more effective.

Project post-mortems look back after a project finished, seeking lessons learned in the hope of applying them to future projects or spreading them across an organization. Conducting post-mortems at the end of the project offers no chance to actually apply these learnings to the project itself. Agile Retrospectives differ in their intent by seeking

change while a project is in flight, where actions have more impact than they would at the end. These meetings create an opportunity for teams to look for opportunities for change though still rely on the people and organization to commit to those changes.

When an organization reaches its goals, it's time to return the metrics used to achieve it. Remember that if organizations accomplished their goals instantaneously, there would never be a need for a metric. Defining, tracking and monitoring and interpreting metrics takes time and resources that could be better spent against new goals. Organizations need to drop metrics that are no longer relevant, instead of holding on to all the metrics they are used to collecting. With an appropriate use of metrics, understanding what metrics to retire will be easy because those metrics have been explicitly linked to the goal and constant monitoring of trends during periods encourage a continuous review of the state of the end goal.

You can look for some symptoms for potentially out-dated metrics by asking people, "Why do we need to collect this number?" A terrible response might include, "That's the way we've always done it," or worse yet, "It's our policy." This question does not necessarily discriminate between poorly explained goals, or out-dated metrics so it probably takes a little bit more digging. It is management's responsibility to ensure that an organization's time is not spent needlessly gathering, maintaining unnecessary metrics.

Conclusion

Metrics have a purpose and a place in organizations and teams. They cannot be used as a substitute for thinking. Nor can organizations think management by numbers is enough for effective software delivery. Organizations must be vigilant against the undesirable behaviors that emerge due to the inappropriate use of metrics. Double loop learning helps us understand focusing on the individual to behave differently cannot exist until the organization learns a more appropriate use for metrics.

With the appropriate use of metrics, organizations link each measure back to a well-articulated goal that everyone understands. The measure chosen to monitor progress must be decoupled from the goal, and challenging each metric's relevance welcomed as time passes. Organizations using metrics more appropriately understand the value in watching the trends, monitoring in smaller periods in order to understand individual, management and organizational influences. A better use also means frequent inspecting and adapting these influences to ensure trends accelerate, decelerate and reverse in the context of an end goal that is constantly evaluated for fitness. The most appropriate use of metrics also means understanding when measures are no longer relevant, replacing them, or dropping them as progress is made towards the goal and the environment changes around it.

metrics productivity project planning technical leadership

Related articles

You might find the following articles on the use and mis-use of metrics interesting to read:

- Gaming Incentives by Esther Derby - A reflection on how people manipulate situations to maximise their incentivization scheme.
- Vanity Metrics vs. Actionable Metrics by Eric Ries - Lean Startup evangelist, Eric Ries describes how to make metrics more actionable and the dangers of measurement for the sake of measurement.
- Velocity is Killing Agilty by Jim Highsmith - A highly relevant article describing how companies misappropriate velocity as a metric.

Footnotes

1: Chris Arygris & Donald A. Schön describes the concepts of single-loop and double-loop learning in their book Organizational Learning: A theory of action perspective.

2: As described in User Stories Applied: For Agile Software Development

Significant Revisions

19 February 2013: First published