silverthread

How honest are your software measures?

Process measures are easy to spin and game. Product measures are more honest.

Software systems need more effective measurements.
Software measures in most system development enterprises seem to reflect the disingenuous spin of politicians more than the matter-of-fact observations of engineers. That statement is probably an exaggeration for some people but the cold hard truth for many others. Although politicians have a well-deserved reputation for subjective spin and a track record for under-delivering on committed improvements, many software delivery enterprises do too.

Systems and software industry leaders are justifiably cynical because our experience with software productivity improvement — internally from our teams as well as externally from our vendors — is plagued by subjective spin. While our intentions with measurement are usually honorable, our know-how is still immature and compartmentalized. It is difficult, demotivating, and inefficient to sell and deliver in a market where there is a default bias of uncertainty and distrust.

Achieving a successful measurement transformation is daunting at small scale and becomes even more challenging in larger enterprises. Strong cultural resistance and cynicism resulting from years of counterproductive measurement practices typically interfere. There is also a broad spectrum of context-specific differences in defining progress and quality. As a result, few standard measurement units, benchmarks of goodness, and measurement practices have gained broad consensus.

Why are conventional software measurements so ineffective?
The weaknesses of conventional wisdom in software measurement can be captured in a few probing questions:

- *Are you emphasizing the measures of supporting artifacts in the process pipeline (often noisy) over measures of the design/code/test artifacts in the product pipeline (mostly signal)?* If you are, the result will be more guesswork and gamesmanship, and less predictable outcomes.
- *Are you measuring things that are useful to management but not to practitioners (or vice versa)?* Either approach can erode trust between stakeholders.
- *Are you measuring individual practitioner performance rather than team outcomes?* Software delivery is a team sport, and this approach can demotivate people and encourage individual gamesmanship.
- *Are you emphasizing specific targets (position) rather than trends (velocity)?* This can lead to short-sighted decisions and downstream surprises.
- *Are you treating measurement targets as static values rather than evolving distributions of probable outcomes?* The result can be misleading confidence in forecasted targets.

These measurement norms are anti-patterns that reflect dishonest, or less honest, behaviors. Software cost, schedule, and quality targets are negotiated forecasts. They should be captured as probability distributions of possible outcomes. The net result of a typical mix of the anti-patterns listed above is excessive measurement noise that drowns out the important signals and results in significant overhead, rework, and waste. Measurement is typically demanded from the top down and rejected (or even worse, gamed) from the bottom up. In the middle, where project leadership must connect technical execution with expected business outcomes, a lot of assumptions, kludges, and spin are created to appease all the constituencies. The usual result is high overhead, a lot of unproductive churn, distrust, and waste.

You probably already use traditional project management measures like those shown on the right-hand side of Figure 1. These process measures provide only half of the insight you need, and most of this insight is subjective or indirectly indicative of progress and quality because it is derived from supporting artifacts. Design quality and code quality are the other half, the *more honest* and important product measures extracted from the primary artifacts that teams need to steer software outcomes more objectively and predictably.

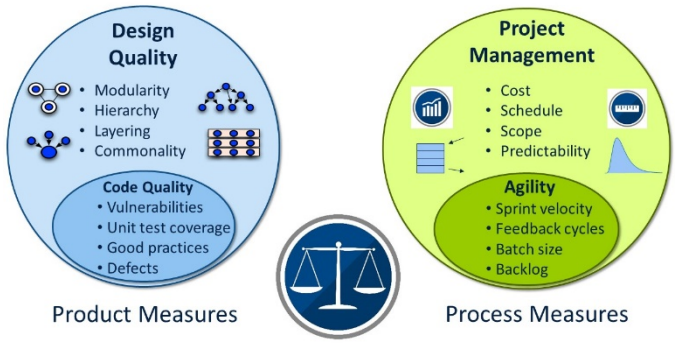

Figure 1: Product measures complement process measures.

Building on lean measurement techniques, we need to quantify *work* in progress, not *activities* in process. This means measuring code in the product pipeline to supplement the less honest measures of artifacts in the process pipeline. The direct measures of the code base are more honest and trustworthy mechanisms for steering software delivery projects. Some enterprises do measure *code quality* of the code subjected to unit testing. This is not enough. Decades of Harvard/MIT research and field experience provide us with strong evidence that the *design quality* of the code base subjected to integration testing correlates better with economic outcomes.

Measurement lessons we have learned
Exploiting practices from positive experiences and avoiding the anti-patterns lead us to a new starting point, a set of guiding principles for a more honest measurement approach. These principles are based on decades of lessons learned in research and field practice.

1. Measure the dynamic characteristics of the product pipeline, not the process pipeline. Direct measures of the code/test base are objective facts and mostly signal. Measures of the process and other supporting artifacts are indirect indicators

and more subjective guesses. They are noisier and easier to game.

2. Better steering involves tradeoffs among competing dimensions. Efficiency (product agility and process agility) and effectiveness (design quality and user-perceived value) must be communicated transparently in objective measures to provide objective tradeoffs and steering decisions.
3. Forecasted measures should be captured as distributions of probable outcomes. It is fine to communicate an expected value, but a more honest exchange would include a discussion of the variance. Why are you confident (narrow variance)? Why are you uncertain (wide variance)?
4. Measurement precision and fidelity should improve along the life cycle. As validated learning increases and uncertainties are resolved, more precision is appropriate.
5. Software agility is as much a function of design (architecture) as it is of process. The most important characteristic of software is that it is "soft." The faster and easier that software is to change, the faster and easier it is to achieve any of its other required characteristics.
6. Metric collection should be automated. Automation eliminates manual overhead and improves consistency.

Improved measurement of *complexity* and *design quality* can increase trust in software delivery. Increasing trust enables leaner production by reducing sources of overhead, unnecessary rework, and waste. Trust is the currency of lean engineering efficiency. The foundations of modern agile methods and DevOps principles revolve around delivering software quickly, measuring "velocity," using smaller batch sizes, improving feedback cycles, and reducing waste. These techniques implicitly and explicitly reduce uncertainty by measuring the primary artifacts of design, code, and test.

Why care about more honest measurement?

There is a hunger for better measurement in software delivery initiatives across all levels of the enterprise. Demand for more objective steering and software delivery analytics is coming from three distinct constituencies with different needs:

1. Enterprise leadership. As software becomes the primary differentiator within most enterprises, a better understanding of software-based progress and quality trends has become paramount to better business predictability.
2. Practitioners. Development and operations professionals will not tolerate the overhead and waste of measuring noisy sources, like the process pipeline, and supporting artifacts. They know that true progress and quality insight come directly from the dynamics of the product pipeline (the code and test base), its change trends, and usage feedback.
3. Middle management. Team leaders, architects, project managers, development managers, and operations managers have the most acute transformation challenge. The middle management job, that of translating technical progress of practitioners into improved economic outcomes demanded by the enterprise, is where the cultural inertia is most entrenched.

Persuading practitioners and executives that improved measures are desirable is usually straightforward. They can easily appreciate the impact of improved accountability and insight on their daily responsibilities. Middle managers, however, carry the burden for the iceberg floating around in most software delivery cultures. They know that moving to more honest and accurate exchanges of information is culturally dangerous in a low-trust environment. Consequently, measurement improvements are best initiated by winning the hearts and minds of middle managers.

Creating shared measures for all constituencies

In most enterprises, measurement and good governance compete with practitioner freedom. Here are two recurring observations from such cultures:

1. Where there is a perception of predictable governance, management morale is positive but practitioners feel choked by high overhead and repetitive manual reporting.
2. Where there is a perception of agility, practitioner morale is positive but management feels out of control with rapidly changing baselines.

Effective measurement approaches must deliver the critical *quid pro quo* illustrated in Figure 2: less overhead and more agility for practitioners, and predictably better economic outcomes for all governance stakeholders. When practitioners and leadership are relying on the same measures, trust will grow.

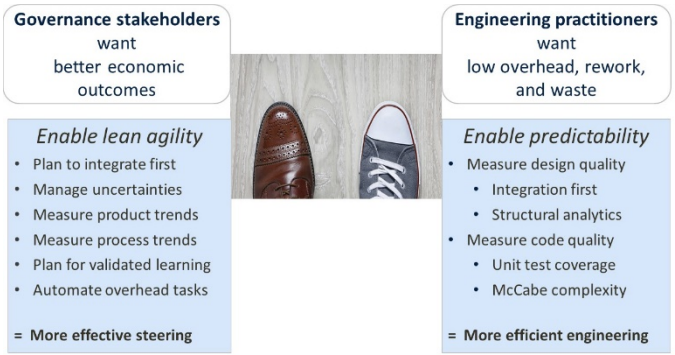

Figure 2: The critical quid pro quo

Stephen Covey coined the term "the speed of trust" to illuminate the necessary ingredient in reducing overhead and improving efficiency and effectiveness: trust. The speed of trust can also be appreciated by its logical opposite: the slowness of distrust. Overhead activities in most enterprises are directly proportional to the amount of distrust. Complexity leads to uncertainty which leads to distrust which leads to more waste and overhead activity that practitioners hate. More honest measurement is the foundation for establishing a higher trust environment among software stakeholders.

Contact Us

Silverthreads's mission is to advance the state of software measurement practice by quantifying complexity and design quality. Our measurement know-how can establish a more trustworthy foundation for improving software economics.
http://silverthreadinc.com