Economic Governance of Software Delivery

Murray Cantor
Distinguished Engineer, IBM

Walker Royce
Chief Software Economist, IBM

***Abstract*–Agility without objective governance cannot scale, and governance without agility cannot compete. Agile methods are mainstream, and software enterprises are adopting these practices in diverse delivery contexts and at enterprise scale. IBM's broad industry experience with agile transformations and deep internal know-how point to three key principles to deliver sustained measurable improvements in software business outcomes with higher confidence: Measure change and minimize overhead, steer using economic governance, and plan and predict using Bayesian analytics. Applying these three principles in context is the crux of measured improvement in business agility and continuous delivery of smarter software-intensive systems. This paper describes more meaningful measurement and prediction foundations for economic governance.**

Index Terms–economic governance, measuring agility, minimizing overhead, accelerating software delivery, Bayesian analytics, steering leadership.

The principles of agile software development are more than 10 years old. Many practical ideas have been piloted, practiced, and evolved. Agile practices are now neither novel nor extreme. What differentiates the enterprises that significantly improve software productivity from those that flounder? One significant discriminator an appropriate governance model that complements the dynamics of agile principles and practices. *Economic governance* is a foundation for quantified planning, decision-making, and measuring that resolves uncertainty earlier and unifies constituencies on managing a shared set of expected target outcomes.

Successfully delivering software in a predictable and profitable manner requires an agile leadership style that *expects* an emerging mixture of discovery, production, assessment, and refactoring. The software industry has been advancing agile methods and deploying agile practices for two decades, and we have observed the natural complement of agile development techniques with the dynamic steering principles of economic governance. References [1] and [2] provide good discussions of agile methods.

Modern agile practices apply to most software development endeavors where achieving a better economic outcome is the primary objective. Agility means *quick to react* or changing directions freely and fluidly, thereby encouraging thoughtful refactoring when necessary. To reason about agility in software systems, we need to quantify change trends and measurably improve the efficiency of change cycles. True agility in software delivery translates into business flexibility. Organizations and projects that can exploit agility with economic governance have an advantage. When changes are less costly to implement, a project is more likely to increase the number of changes, thereby improving quality, efficiency, and timeliness.

True agility means that life-cycle changes become more predictable and straightforward over time. Agility is best measured by quantifying the costs of change over time. When you find teams that have improved their turnaround time for software changes from a few weeks to a few days, you know they have become more agile. Instead of being mired in late scrap and rework, and consumed playing defense, you can go on the offensive by innovating more, adding more features or more quality, improving performance, or delivering earlier.

Transforming from conventional engineering governance to economic governance will change the principles and perspectives that are driving the technical and business leadership. An

engineering orientation focuses on static targets, planned activities, and deterministic predictions to drive the development process, whereas an economic orientation steers project priorities and results based on the dynamically changing predictions of the probable outcomes of the development process.

Software delivery: Economics trumps engineering

Successful software outcomes are highly dependent on continuous negotiations, accurate predictions, value judgments, innovations, team collaboration, architects, agility, market conditions, and user demand. Success is much less dependent on quality of contracts, Gantt charts, critical path schedules, earned value measurement, laws of physics, material properties, mature building codes, and certified engineers. *Software delivery governance is more a discipline of economics than it is of engineering. It is a nondeterministic endeavor with much more inherent uncertainty. Entrepreneurial leadership principles trump our conventional engineering management doctrine.*

Engineering endeavors are generally governed by well-understood laws of physics and properties of materials, as well as centuries of precedent experience. High-uncertainty engineering efforts are still attempted, such as capping the Gulf of Mexico oil leak in 2010, but these are the exceptions. Software delivery, by comparison, is a discipline dominated by human creativity, market forecasting, value judgments, and uncertainty levels similar to economic endeavors such as movie production and venture capital management.

Modern software delivery is a creative discipline more akin to making movies, producing a play, or writing a book [3, 4]. Consider these three observations:

1. Most software activities have no laws of physics or properties of materials to constrain their problems or solutions. They are bound only by human imagination and economic constraints.
2. In a software project, you can change almost anything at any time: plans, people, funding, requirements, designs, and tests. Requirements—probably the most misused word in our industry—rarely describe anything that is truly *required*. Nearly everything is negotiable.
3. Quality metrics for software products have few accepted benchmarks. With the possible exception of reliability, most aspects of quality, such as responsiveness, maintainability, and usability, entail vast uncertainty and remain largely measured subjectively, through the *eyes of their beholders* in the user community.

Software project managers, movie producers, and many of today's entrepreneurs create a unique and complex web of intellectual property bounded only by an abstract vision and human creativity. These endeavors experience a lower success rate than mature engineering enterprises. Software engineering discipline has its place, but software governance techniques must steer through far greater levels of uncertainty to deliver better economic outcomes.

This mindset change is well-articulated by Eric Ries in *The Lean Startup* [5]. *The Lean Startup* approach begins by capturing the business case as a clear hypothesis (a prediction of the value proposition) and then tests that prediction through a sequence of experiments that empirically validate a strategy. When choosing among the many assertions in a business plan, it makes sense to test the riskiest assumptions first because they result in the most significant reductions in uncertainty. A minimum viable product (MVP) constitutes the critical first experiment.

Transforming to economic governance

Software enterprises face a significant challenge to transform their culture from the deterministic world of conventional engineering governance to the probabilistic mindset inherent in economic governance. This is a quantum leap for most stakeholders from static targets to dynamic targets, from managing certainty to managing uncertainty, and from deterministic decisions to probabilistic decisions. Table 1 illuminates some (purposely stark) shifts in governance mindset. To highlight the transformation priorities, it tends toward worst-case scenarios for engineering governance and best-case scenarios for economic governance. Over the last 2 decades, we (the authors) have worked with hundreds of projects mired in engineering governance and perhaps twenty projects that succeeded with the patterns of economic governance.

Conventional engineering project management techniques assume little uncertainty in their requirements and exploit mature precedents for production and deployment. Software projects managed with such engineering governance models typically uncover stifling, malignant changes late in the lifecycle and spend 40% or more of their effort consumed in late scrap and rework [6]. The iron law of traditional software engineering is this: The later you are in the lifecycle, the more expensive things are to fix. If you are experiencing this iron law, you are probably anchored in a conventional waterfall-model process. Your process may be mature, but it might be more accurately described as geriatric.

Table 1. Conventional engineering governance vs. modern economic governance

Engineering Governance	Economic Governance
Distinct handoff from development team to maintenance team	Common process, platform, and team for development and maintenance
Distinct and sequential activities: requirements to design to code to test	Continuous delivery of demonstrable outcomes with ever-increasing value
Role-specific processes and tools	Collaborative platform of integrated, web-based tools and practices
Early false precision in plans and requirements; focus on the well-understood elements first	Evolving, honest precision as uncertainties are resolved; focus on the big uncertainties first
A static plan that is a sequence of artifacts: specs, models, presentations, test documents	A dynamic plan that is a sequence of executable, change-controlled releases
Unit test coverage and completeness precedes integration and system level testing.	Integration testing precedes investment in unit test coverage and completeness.
Measurement of artifact production and activity completion	Measurement of progress/quality trends in testable releases
Measurement of achieved values and variance of actual results to planned targets	Quantify uncertainty, negotiate improved targets, and steer to optimize win-win targets.
Deterministic engineering discipline: plan and track mindset based on static targets and deterministic predictions.	**Probabilistic** economic discipline: predict, quantify, and steer based on dynamic targets and objective evidence.

A counterintuitive leap from deterministic decisions to nondeterministic reasoning is necessary to make the transformation to economic governance. This is not easy for most of us. Project managers who are experienced and trained in traditional project management disciplines such as detailed planning, critical-path analysis, and earned value management have a particularly rough

transition. They must move from a world of managing certainty and precision to a world of resolving uncertainty based on imprecise probabilistic judgments and investments.

The crux of economic governance: A parable

Suppose you have a software product that needs to be delivered in 12 months to satisfy a critical business need. As the organizational investor you decide to solicit proposals from two competing project managers. Gerrard Hattrick is a traditional run-of-the-mill software project manager. Jerry is a veteran project manager. Shirley Nimble is a more contemporary software manager. Using empirical estimation models, they both estimate the project should take 11 months. Here is where their approaches diverge.

Jerry is a French-Canadian hockey fan, and with a name like *Hattrick*, he feels compelled to initiate projects with three goals: 1) plan the entire 12 months in detail, 2) specify the requirements precisely, and 3) conduct an early design review with a demonstration of the elements that are easiest to mock up. He wants to demonstrate early progress to keep stakeholders satisfied and to avoid any late changes in scope. He feels confident that he can deliver with a month of *extra* schedule.

Shirley sees things differently. She understands that the model's output is simply the mean of a more complex random variable. She asks to see the full distribution of possible outcomes and a planned sequence of intermediate release content. She wants to go into the project with a 95% chance of delivering on time. Her leadership team shows her the complete distribution illustrated as the *baseline estimate* at the top of Figure 2.

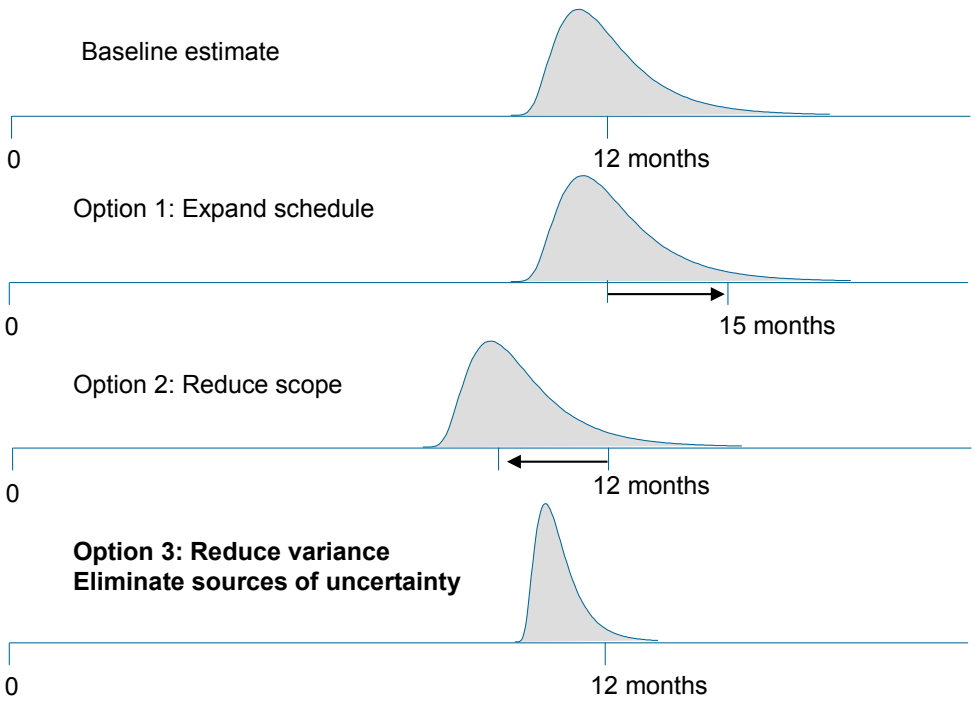

Figure 1. A baseline estimate and three alternatives

Assessing the baseline estimate, they realize that about half of the outcomes will take longer than 12 months, and they have only a 50-50 chance of delivering on time. The reason for this dispersion is the significant uncertainty in the various input parameters, reflecting their team's lack of knowledge about the scope, the design, the plan, and the team capability. Consequently, the variance of the distribution of outcomes is rather wide. There are three paths they can take to ensure that 95% of the outcomes complete within the target date.

1. Option 1: Ask the business to move out the target delivery date to 15 months.
2. Option 2: Ask the business to rescope the work, eliminating some of the features or backing off on quality so that the median schedule estimate moves up by a month or so.
3. Option 3: Explicitly reduce the uncertainties in the scope, the design, the plans, the team, the platform, and the process. This narrows the dispersion in the distribution and increases the probability of delivering within the target date.

The first two options are usually deemed unacceptable by external stakeholders, leaving the third option as the commonly preferred alternative.

Jerry prioritizes his efforts on what he knows. He mostly ignores the big uncertainties, postponing their resolution until after he builds up early project momentum by tackling the straightforward tasks and showing early progress in building the easy parts. At first, Jerry's team will appear to be making progress, but late in the project, one of the big uncertainties will likely explode into a formidable roadblock and there will be a late surprise with significant project delay. Jerry's traditional management instincts may be mature, but when applied to software, a more accurate description might be: *geriatric*.

Shirley takes the smarter and more courageous approach. She proceeds through a series of experiments by building early versions to test the design and gain feedback from target users on progress. These experiments result in *validated learning,* which is an increased understanding (or decreased uncertainty) in the business case, scope, and design of the target system. She resolves the harder challenges first, delivering fewer tangible artifacts early but learning more about what constitutes a successful delivery, thereby increasing the probability of long-term success and decreasing the probability of future scrap and rework. In the long run, Shirley is more likely to deliver the right software on time.

So what's the real crux of change from traditional mindset of Jerry Hattrick and the leaner mindset of Shirley Nimble? Jerry's engineering orientation drives to static targets and deterministic predictions in the face of significant uncertainty. Shirley's economic orientation steers projects with dynamically changing predictions of probable outcomes and an honest discussion of uncertainties. Shirley Nimble is operating like a savvy entrepreneur, striving for and measuring validated learning, as discussed in *The Lean Startup*.

Bayesian reasoning

Setting an expected delivery date or planned resources for any project is a kind of prediction. Many project managers are faced with this dilemma: Although it is impossible to predict the future, that is their job. The best way to improve predictions is to apply *Bayesian reasoning*.

Bayesian reasoning, based on centuries old mathematics, was historically a controversial way of thinking about the nature of probability. It has evolved into accepted best practice, central to all sorts of predictions. For example, in *The Signal and the Noise, Why Most Predictions Fail and*

Some Don't [7], Nate Silver explains how Bayesian reasoning is central to predicting outcomes of elections, economic forecasts, disease propagation, weather, poker, and other societal interests.

Bayesian reasoning requires a change of mindset from the way most of us approach project estimation. We abandon the idea that a project has a fixed duration or a fixed cost. The Bayesian mindset reasons about predictions as a range of possible values called outcomes. Some outcomes are more likely than others, so each is assigned a likelihood value. The basic probability and statistics foundations for Bayesian reasoning are discussed in reference [8].

The challenge for project leadership is to identify an initial distribution for a given random variable, such as time to complete, and then continuously update the distribution as the project proceeds. The changing shape of the distribution provides insight into whether the probability of meeting the schedule target is improving. The initial distribution is called the *prior* and the updated distribution the *posterior*. In Figure 2, the baseline estimate at the top would be the prior and the result after pursuing option 3 at the bottom of the figure would be the posterior. In summary, the process is rather simple:

- Make an initial, but informed guess of a prior distribution.
- Gather evidence.
- Use Bayes' theorem to update a posterior distribution.
- Continue updating the posterior distribution as more evidence becomes available.

One of the important features of this approach is that it converges to the same posterior no matter what prior was used to seed the process. This is important: *The perfect specification and the perfect plan are not necessary to improve predictability*. It is far better to make a reasonable estimate and to start refining that estimate. This is the spirit of iterative and agile development that differentiates it from conventional waterfall model thinking.

More honest predictions

While the most likely outcome may be a rough prediction (usually something close to the mean value), a more *honest* representation of the prediction would be the full probability distribution of possible outcomes. For example, the most likely outcome of 11 months is an accurate portrayal of the expected target date depicted for the upper distribution in Figure 2. However, by expressing how sure we are of that guess—by exposing the variance in the distribution—we are much more honest and transparent in communicating that information to others.

The difference between precision and accuracy is an enlightening lens for insight into honest predictions. Accuracy is a measure of truth and freedom from error. Precision identifies the degree of accuracy, and greater precision implies repeatability or elimination of uncertainty. The pursuit of early precision is alluring but usually serves only to provide a counterproductive façade for portraying illusory progress and quality. Trust among stakeholders erodes as the divergence between reported progress and true progress inevitably reveals itself. Many stakeholders demand early precision and detail because it gives them (false) comfort in the progress achieved. Trust is enhanced with honest communications. Emphasizing accurate estimates with honest qualified precision helps stakeholders engage in more fruitful discussions of the uncertainties remaining and establishes more trust.

Uncertainty can be quantified and tracked by periodically measuring the reduction in variance in the distribution of resource estimates to complete. Each phase of development should reduce

uncertainty in the evolving plans, specifications, and demonstrable releases. At any point in the lifecycle, the precision of the subordinate artifacts, especially the plans and estimates to complete, should be in balance with the evolving precision in understanding.

Covey [9] coined the term *the speed of trust* to illuminate the necessary element in reducing overhead and improving efficiency: 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. Some distrust is healthy and necessary, because humans make errors. However, when overhead activities are overly burdensome (that is, where the value achieved is out of balance with the cost and inconvenience), the system becomes inefficient. More honest communications improves trust among stakeholders. Deterministic planning kills trust, because it doesn't match the reality and uncertainties of the software development world. Communicating plans and targets probabilistically and explicitly quantifying the uncertainty builds trust because it accurately portrays our understanding of how to reason about the future.

Managing uncertainty using agile practices

All stakeholders must collaborate to converge on moving targets and manage uncertainties, as illustrated in Figure 2. Software schedules, scope, designs, and plans must be treated as predictions with inherent uncertainty and not contracts with implied certainties and requirements. A target date is a prediction, typically the mean of a probability distribution of the release dates. Requirement specifications and models are evolving predictions of the negotiated need. They emerge from a coarse vision with a wide variance to a precise, testable specification with relative certainty. Nearly everything is negotiable early in the lifecycle. Operational characteristics remain negotiable as tradeoffs evolve from speculative debates to factual decisions. A project plan can be captured as the predicted set of state changes en route to the desired end state. Precision in these life-cycle artifacts should be added commensurate with the uncertainties that have been resolved.

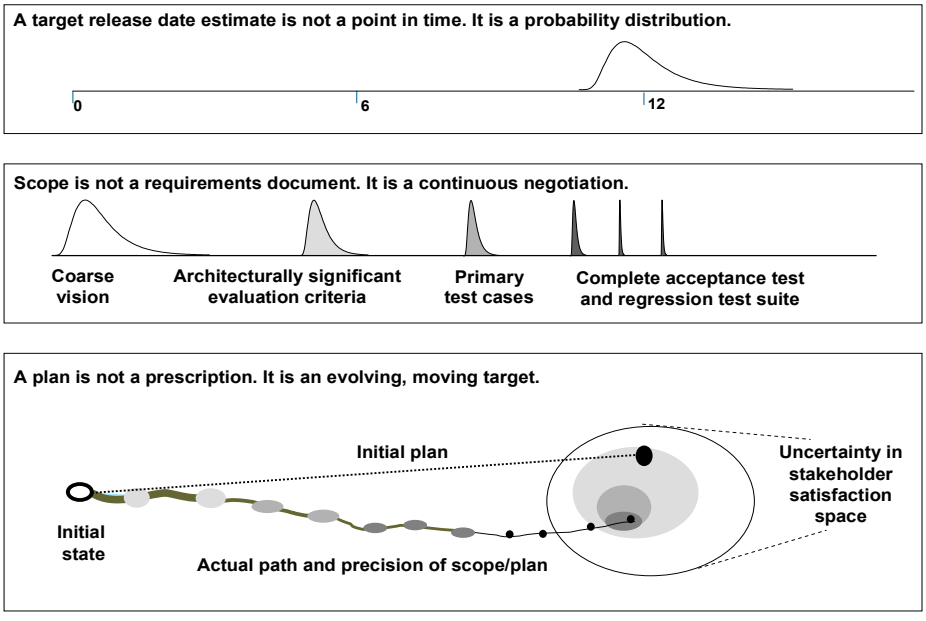

Figure 2. Important steering perspectives to manage uncertainty

Iterative and agile development methods have emerged organically from diverse software development communities to improve the navigation through uncertainty. This steering requires measured improvement with dynamic controls, instrumentation, and intermediate checkpoints that permit stakeholders to assess what they have achieved so far (their as-is situation), what perturbations they should make to the target objectives (their predicted to-be situation), and how to refactor what they have achieved to adjust those targets in the most economical way (the roadmap forward). The key results of these modern, agile delivery principles are reduced overhead and a significant reduction (perhaps as high as 50%) in scrap and rework [6, 10].

To improve the predictability of outcomes, we need to integrate measurement and agility so they are complementary objectives and not competitive forces. Management must empower practitioners with more freedom to change and a platform that automates overhead tasks. In exchange, practitioners must provide management with improved steering mechanisms through transparent progress and control measures that better correlate to economic outcomes.

Applying Bayesian reasoning for improved predictability

A closer look at Shirley Nimble's situation and activities shows that within her resource constraints, she has agreed to deliver a mobile application in 12 months. Shirley plans to use an agile process, decomposing the features into work items and assigning those work items to a sequence of builds. Shirley treats the time to complete as the random variable of one outcome of interest. Before she committed to the project, she developed an initial distribution of the time to complete the project. With this distribution, she had a view of the probability of achieving the goal.

Her team estimated the level of effort required for each feature as simple random variables represented as triangular distributions. In particular, for each feature's level of effort, the leadership team agreed on three values:

- The low (best case): assumes all the stars align and the feature comes together easily to meet requirements
- The high (worse case): assumes that Murphy's law dominates the project circumstances with unexpected challenges and obstacles
- The nominal (most likely): assumes the level of effort has the expected mix of good fortune and bad luck

There is no chance that the level of effort will be less than the best case or more than the worst case, so the distributions are set to be zero below the low value and above the high value.

In Bayesian reasoning, this technique provides the subject matter experts a means of arriving at an honest prior, based on current information and informed belief. If the difference between the low and high of the distribution of a feature is large, then the team is expressing its uncertainty of the effort required to deliver the feature. This gives Shirley's team greater opportunity to resolve the uncertainties early, progressively de-risking the project.

With this prior estimate in place, Shirley had an idea of how likely it was that she could meet the commitment, and she negotiated the content. Through some analysis, she found that one of the relatively uncertain features was more of a nice-to-have than a must-have and added considerably more risk than value. So she negotiated that feature out of scope for a firmer commitment to an earlier delivery in 11 months.

As they complete the work items and plan items, they update the time-to-complete random variables based on increased understanding of progress and quality. Completed work and actual team performance (facts), combined with updated plans-to-complete (estimates), are used to update the time-to-complete distribution (honest predictions) with sound mathematics.

With these periodic predictions, Shirley can discuss with her internal team and external stakeholders whether the odds of meeting the commitment are improving (as they should) or not. If the odds are degrading, she can intervene earlier by further managing content, adjusting resources, or steering the project in some other way toward a better outcome. With a quantified and objective view of the project trends, all stakeholders can have a more honest and trustworthy discussion about how best to proceed.

Conclusions

IBM's broad industry experience with 100s of projects and 10 years of internal agile transformation experience point to three key principles to deliver sustained measureable improvements in software business outcomes with higher confidence:

1. **Measure** change and minimize overhead.
2. **Steer** using economic governance.
3. **Plan and predict** using Bayesian analytics.

Measure. Continuous measurement in executable software baselines is a cornerstone of agility. Measurements must illuminate the progress and quality indicators needed to steer projects to more successful outcomes. The agile principles of continuous integration and test-driven development lay a strong foundation for measured improvement. The core metrics for steering can be extracted from the release baselines undergoing continuous integration testing. These metrics quantify the validated learning for more honest reduction of uncertainty and trustworthy assessment of progress and quality.

Steer. Steering projects with economic governance demands a different outlook on leadership priorities:

- Manage target costs and schedules as a narrowing distribution of outcomes
- Predict outcomes using Bayesian reasoning and ever-improving information
- Optimize quality as a narrowing distribution of defect classes and density
- Communicate progress and quality trends transparently and honestly

Plan and predict. Software estimates, proposals, and plans are essential predictions that represent the primary information exchanges among governance stakeholders. Trust is earned when integrity and performance are combined. Integrity is improved with more honest predictions that quantify uncertainties. Performance is improved by measurably reducing uncertainty early and continuously negotiating and steering.

Transforming. In most of today's software engineering cultures, which are anchored in deterministic planning, middle management governance competes with practitioner freedom. Practitioners must provide management with improved steering mechanisms such as progress and control measures, automated instrumentation, and real-time development analytics. Management must give practitioners more freedom to innovate through automation of measurement,

traceability, progress reporting, documentation, and change propagation. The platform of process know-how and automation must deliver this critical *quid pro quo*:

Practitioners must demand that overhead tasks be minimized or automated. Management stakeholders, on the other hand, need more insightful measures and a feedback control loop to steer progress and quality with better predictability of business outcomes. Trust is the secret to achieving this *measurement-for-freedom* exchange.

There are probably more books on agile methods than successful projects with well-documented agile results. Writing a book on agility or project management is easy. Managing a real project where you must steer through a minefield of uncertainties is hard. We should all place increased emphasis on publishing measured improvement case studies. They are critical to accelerating the innovation delivered in software.

References

[1] Ambler, S.W., and M.J. Lines. *Disciplined Agile Delivery: A Practitioner's Guide to Agile Solution Delivery in the Enterprise*. IBM Press, 2012.
[2] Kennaley, M. *SDLC 3.0, Beyond a Tacit Understanding of Agile*. Fourth Medium Press, 2010.
[3] Royce, Walker. "Successful Software Management Style: Steering and Balance." IEEE Software 22(5):40-47 (September/October 2005).
[4] Austin, Rob, and Lee Devin. *Artful Making*. Prentice Hall, 2003.
[5] Ries, Eric. *The Lean Startup*. Crown Business, 2011.
[6] Royce, Walker E. *Software Project Management*. Addison-Wesley, 1998.
[7] Silver, Nate. *The Signal and the Noise, Why Most Predictions Fail and Some Don't*. Penguin Press, 2012.
[8] Cantor, Murray. *Filling in the Blanks, The Math behind Nate Silver's The Signal and the Noise*. IBM Developer Works, 2013.
[9] Covey, S.M.R. *The Speed of Trust: The One Thing That Changes Everything.* Free Press, 2006.
[10] Royce, E. "Measuring Agility and Architectural Integrity." Int J Softw Informatica 5(3) (2011).