Dragon Scales: 50 Teams Scrumming

Implementing Adaptive Project Management Practices at Scale

Abstract |
Product portfolios can easily scale to 50 teams or more in meeting large organizations' needs. Large portfolios with strong foundations are derived through values-based leadership. The technique links corporate and individual values to scientific principles. Scientific principles inform us that change is constant and therefore adaptation defines good practices. Values-based leadership's agile practices take root, thrive, and adapt at the pace of business change.

The three-hundred software engineers considered herein innovated within a portfolio of 18,000 colleagues. Their agile, adaptive product development practices continue to evolve from plan-driven provenance. Leveraging agile practices at the portfolio, program, and project level continually unleashes innovation, quality, and throughput of value. Though contextualized in terms of software product development in the 2010s with Scrum, the message of innovation through values-based adoption of scientific principles is timeless and framework unallied. Implementation of practices observant of values and principles endures as a way to deliver the best products regardless of toolset.

Key words |
Scrum, Values, Principles, Practices, Agile, Change, Scaled agile

Introduction

Motivating several hundred software developers to adopt agile seldom occurs based on the project manager's legitimate, reward, expert, referent, or punishment power. A personal, internal motivating force is required. Appeals must be made based on common values of the employee and organization. Software developers value creativity, excellence, and openness. Openness in the 2010s became the developer's mantra. Open source software products rose to prominence in product development.

Product development at scale involves numerous groups of products. This aligns to the essential definition of a portfolio, a group of programs and projects (PMI, 2013). The portfolio alluded to herein consists of 18,000 software engineers and a sprawling product life cycle management (PLM) platform. Dozens of products in this framework have a thirty-year history of innovation benefiting product engineers across the globe.

The global program designs interfaces for any size device aligned to the specific needs of the product engineer. Then modern software development called for HTML5 and CSS3 technologies to deliver the suite. The suite always focuses on the customer. Agile approaches were selected, given the complex nature of the endeavor and need for customer feedback.

The program delivers in a domain of emergence. Often answers emerge only after the customer sees the product. It can only be known in hindsight if the product meets the needs of several thousand customers, using dozens of platforms. Delivering small units of work for quick review and feedback proves to be economical and essential. The endeavor is more unpredictable than

Originally published as part of the PMI® Global Congress 2016—North America proceedings.

Global Congress 2016
North America

predictable. Agile provides the only tenable approach offering time to react to feedback.

The timely selection of which interface the customer wants next determines product success. The next things built must follow market demand. Smaller incremental builds generate new revenues and harvest market feedback. The risk of building the wrong product is limited to one smaller incremental build.

Building in smaller sets forces frequent and often painful integration of software. To engineers with a thirty-year history of building software every 18 months, this is an epic adventure requiring a rallying cause, steady direction, focused processes, and a wholly intentional leadership strategy. It began with common values. *Why* this should be done.

Values, Principles, and Practices

If you want to go fast go alone, if you want to go far go together. – African Proverb

There are dozens of agile frameworks. They can be described in terms of their values, principles, and practices (Shalloway 2009). Software engineers are intellectual, driven, and enjoy quality work. So they have affinity for values aligned to these tendencies. Well-wrought frameworks align to the team's values. Examples abound. Scrum is silent on the matter of documentation. Extreme Programming (XP) explicitly calls for documentation. Some software engineers are comfortable proceeding without documentation. Some global teams carry extra communications burdens needing documentation for people in other time zones, countries, and cultures. Values guide the selection of such practices. Global teams place a premium on communications. They often prize responsibly documenting architectural designs to share company wide.

Mission, vison, and values are lingua franca in large corporations. Of these three facets, values form the communicative edge. Values explain why we do what we do. We write extra documentation in difficult circumstances to make sure we respect every individual. Values give people the basis to act according to principles. Values are a moral catalyst.

But what is a value? Courage is a great example. The first of all virtues, courage allows us to take a risk, to speak truth to power, to make an estimate, and to make a commitment. It's essential to any agile framework. The power of values-based leadership is well-stated in *The Scrum Guide*™,

> When the values of commitment, courage, focus, openness and respect are embodied and lived by the Scrum Team, the Scrum pillars of transparency, inspection, and adaptation come to life and build trust for everyone. The Scrum Team members learn and explore those values as they work with the Scrum events, roles and artifacts.
>
> (Schwaber & Sutherland 2016, p. 4)

Where there is fear, there must be courage. Software engineers dread the project manager's question: "When will this be done?" The software engineer values quality. The project manager values predictability. Tension and fear can ensue.

In the core cadre of a Scrum team, we also have business analysts, testers, user experience (UX) professionals, and technical writers. Business analysts value completeness. Testers value certainty. UX people value design. Technical writers value clarity and completeness. In all cases, people need their values understood to fully embrace the program. People with detachment from values become a serious anti-pattern for success. People unmoored from values perform nominally and often against immutable principles governing outcomes.

These principles act like forces of nature, much like gravity. Release a heavy object toward your foot, expect pain. Deploy a heavily plan-driven project management framework in a domain of emergence, expect project death marches, overrun budgets and schedules, and suboptimal products. Dozens of agile frameworks identify principles at play in the domain of emergence.

Not everything is emergent. Not everything is agile. There are of domains of commonality where the right thing to do is obvious. Projects repeating for the hundredth time with obvious correct answers

Originally published as part of the PMI® Global Congress 2016—North America proceedings.

benefit little from adaptive, agile overhead. If the principles at play indicate a project is highly reproducible, the right answers are obvious, and best practices clearly exist, the program would benefit from a plan-driven approach. Such frameworks include Six Sigma, Lean, or Kanban in the agile gamut. Success requires review and diagnosis of the principles at play. The basic question is: "Will this program be operating in a domain of emergence?" Product development does.

Success requires the courage to call out the principles at play, often in the face of dogmatic policy, and support the correct framework. Plan-driven practices will certainly fail in a domain of emergence. Conversely, adaptive, agile processes applied to a simple program create chaos and waste. Ironically, these are the anti-patterns agile practices seek to avoid.

What is an agile practice? Agile practices include things like exploratory testing, the daily stand-up meeting, and UX design documents. Of the dozens of agile frameworks, the common thread is that practices intentionally address principles at play.

For example, the Extreme Programming (XP) practice of having a daily stand-up meeting (Scrum calls this the daily Scrum) supports the principle of inspection and adaption. Inspection and adaption is proven to deliver better results. A team that frequently inspects work always outperforms a team that doesn't. Look to the automotive industry for proof.

Inspection and adaptation is proven to work in multiple industries. The true question is: "Will the team value the practice?" The answer reveals itself through conversation about values, principles, and practices. Portfolio, program, and project managers accelerate the throughput of business value and decrease the cycle time to deliver value with a values-based conversation regarding any new or existing portfolio practice.

Values, principles, and practices shift societally, scientifically, and experientially. An agile program, by design, never completes. It is always adapting, with change being permanent. This adaptation is an intentional, continual selection process. A stable core of road tested values, principles, and practices is foundational.

To initiate a large program, select a core agile framework. Scrum dominates the market presently. It is a simple framework with five required events. Scrum is elegant in compactness. The core values are courage, openness, respect, commitment, and focus—concepts easily embraced. The principles are transparency, inspection, and adaption. Principles empirically proven to outperform since the days of Deming in Japan. The five practices are the sprint, the daily Scrum, the sprint planning meeting, the sprint review, and the retrospective. These practices, road tested by thousands of teams across the globe, form a bare-bones framework based on values.

Through values, the team takes courage to do the right thing. Through openness, the current state of affairs is available for adaptation. Through focus, each person gets to work on the most important thing first with concentration. Through respect, teams come to know that when they make a commitment it is understood to be a promise. They will do their best in a fluid enterprise that will change constantly.

Agile frameworks are described in terms of these values, principles, and practices. Values are a fine entry point to understanding a framework and judging if it will fit the situation.

In the case of this large software engineering endeavor, the Scrum values aligned 100% to the corporation's stated values. Scrum aligns well to almost any program endeavor. This is why it dominates the market as a framework of choice. However, the fit may not be 100%.

Software developers building systems with millions of lines of code have additional values. Though they almost universally ascribe to Scrum values, the need for simplicity is dominant. Simplicity is required to get programs to run on devices ranging from large engineering workstations to smartphones. Simplicity is required to integrate millions of lines of code efficiently.

The efficacy of frequent feedback is valued and required. Feedback requires code to be frequently integrated across the platform landscape to minimize rework and ultimately limit bad investment in unmarketable products. The risk of producing a "wrong product" is limited to a single

Originally published as part of the PMI® Global Congress 2016—North America proceedings.

short iteration. It's necessary to be open and communicate issues as soon as they are spotted.

XP espouses communications, feedback, simplicity, and testing as core values. All of these values may be incrementally adopted in the large software product development program. This values mélange lets software engineers know that there are important additional principles and practices required. To build a great product, the broader team will work within the core Scrum framework, adding XP practices as principles observed to be in play. This is essential modification of the process.

XP is the most detailed framework. XP identifies at least 10 more principles than Scrums, including the concept of quality work. Quality work observes that cost increases uncontrollably if quality is not built in. The cost of poor quality is the internal failure cost plus the external failure cost. In product development, the internal cost is counted in employee attrition, wasted effort, failure to deliver business value, and poor morale. External costs like losing market share and dissatisfied customers accelerate a product's end of life.

Understanding the principle that risk creates waste, XP offers more than thirty practices (to Scrum's five) to deliver quality work.

The XP practices are categorized into thinking, collaborative, releasing, planning, and development practices. In aggregate, the thirty practices cover everything for small teams. Situationally, each of these thirty are the best solution for some challenges faced. The majority of software product development teams frequently need specific, individual XP practices to improve releases, development, or any of the other categories.

Several hundred people working across multiple time zones will necessarily have different learning curves. Adopting XP's numerous practices in a coordinated fashion is certainly a larger task than adopting Scrum. Getting onto the Scrum learning curve and adopting XP values piecemeal over time offers a smoother glide path. A small team might be able to efficiently adopt XP in one fell swoop. A large-scaled program should best tailor the Scrum framework incrementally, adding good practices, XP, and others.

The deduction here is that a few dozen agile methodologies offer practices. No one framework is likely to cover all situations. By definition, agile is never done, incrementally improving practices is a core tenet. At large-scale, an incremental adoption plan of select practices is more practical. Scrum provides a solid core for scaling endeavors but lacks (probably intentionally) all facets needed for all situations. Maximum leverage is attained by identifying a contender list of relevant practices. This list should be prioritized by program stakeholders. An intentional progression follows. Practices are layered onto the existing core and empirically observed for improvement. Practices that fail to improve operations are abandoned and others are tried, measured, and considered.

Agile is continually transitional. Adaptation is a permanent feature. Through the lens of values, principles, and practices, the organization focuses on continuous improvement driven by ever evolving techniques and empiricism. As such, the portfolio, program, and team constantly devise new high-leverage practices.

Leverage

Give me a lever long enough and a fulcrum on which to place it, and I shall move the world.
– Archimedes

From the C-Suite to the team, commit to scaled agility as a means and not as an end, to a journey and not to a destination. The task is undertaken with ruthless prioritization. Select and execute copasetic practices. If they don't work, move on to others. The optimal goal is to maximize value observing Little's Law on throughput, the Lean Project Management commandment regarding workflow and cited so frequently in agile literature.

The goal is to increase the throughput of business value (e.g., number of stories delivered) while decreasing the cycle time. Cycle time is the time to deliver (e.g., five days to get a story). The need to increase throughput and decrease cycle time is observed in all agile frameworks. Constant adaptation of products and the processes that build them increases throughput.

Originally published as part of the PMI® Global Congress 2016—North America proceedings.

A constant inventory of practices must be prioritized. From this list emerges a set of key transition, portfolio, program, and team-level practices offering high leverage. Continual planning and review of these high-leverage practices creates a road map for scaled program success. Quantify high-leverage activities, set priority for improvements, chaperone their implementation, and set targets for accomplishments to scale well.

There are two key questions to ask about the high-leverage practices. First, on a scale of 1 to 5: "How important is the leverage point to our portfolio?" Second, on that same scale: "How important is it to improve this practice in the next iteration?"

The form of the two key questions above is excellent fundamental data for the Kano survey technique. Kano surveys are ideal for communicating the portfolio's plan for establishing a framework and high-leverage practices (Cohn 2006). The key point of the Kano survey is it not only asks people what to change but also asks them what they want to change. This creates an investment in the transition to scaled agility.

Two-dozen leverage points are listed below. They may serve as a starting point for other portfolios scaling agile. The process of identifying and augmenting a framework with high-leverage practices is universally applicable, though some noted practices are specific to software development.

Transition Leverage

Pace, Managing the Transition, Mentoring the Team

Portfolio Leverage

Budget Cycle, **Sequencing Work, Visibility**, **Requirements, Managing the Work Load**

Program Leverage

Work Iterations, **Program Roles**, **Cadence, Managing Tactical Work**, **Align the Portfolio with Teams**

Team Process Leverage

Size of Stories, Organization of Teams, **Ecosystem**, **Acceptance Test-Driven Delivery (ATDD)**, **Agile Text Matrix, Team Methods, Processes, and Framework**

Team Technical Leverage

Degree of Test Automation, Test First Methods, **Design Approach**, **Integration Cadence**

Context

Agile rejects the heavy methodology of plan-driven or iterative project management. Instead of attempting to outline the roles, actions, and technology to be used for every possible circumstance, agile lets teams innovate. There should be as much structure as is absolutely necessary, and no more.

Scaling agile to the portfolio in the 2010s has seen the emergence of several heavier methodologies (e.g., SAFe, Less, and others). A light framework is better. But circumstances may require a more prescriptive framework. Circumstances to scrutinize when deciding on framework weight include investment and life criticality. In the latter, portfolios investing business-critical capital benefit from more prescriptive portfolio frameworks. For the former, portfolios may be delivering products that lives depend upon. Life-critical products benefit from prescriptions for due diligence.

Prescriptive frameworks are inherently less agile, adaptive. This is potentially the right answer for certain situations. The litmus test, of course, is adaptation itself. Any product intended to endure beyond one release should be built embracing constant change, proscribing a detailed prescription.

Change Is

To improve is to change; to be perfect is to change often. – Winston Churchill

Much of adaptive portfolio, program, and project management's success depends upon people embracing change. The few dozen high-leverage principles described above may represent big changes for organizations. In most cases, it's prudent to anticipate resistance. It, however, is a mistake to assume people will resist change.

People do not resist change; they resist the change process. Many organizations have

Originally published as part of the PMI® Global Congress 2016—North America proceedings.

intentionally designed processes to dampen change velocities and maintain the status quo in a product. This has the impact of shortening the life of a product, but this may be in line with corporate investment strategies. People operating under these considerations will correctly not see the value to constant adaptation. Why change if the marching orders are to sweat the assets?

So, in conclusion, the preamble to any effort at scaling agility is discussion of why adaptation is valued in an organization. If adaptation has not been valued, it is time for a courageous discussion.

About the Author

Andy Burns, PMI-ACP, PMP, is a proud "traditional project manager." Burns is Chief ScrumMaster at one of the largest Product Life Cycle Management (PLM) software companies. Burns Scrums at scale with more than 50 teams across the globe. As a PMI Global Volunteer, Burns mentors the 24 Chapters in Region 4

References

Cohn, M. (2006). Agile estimating and planning. New York, NY: Pearson Education.

Crispin, L. (2009). *Agile testing: A practical guide for testers and agile teams*. New York, NY: Pearson Education.

Larman, C. (2009). *Scaling lean and agile development: Thinking and organizational tools for large scale Scrum*. Boston, MA: Addison-Wesley Educational Publishers Inc.

Martin, R. (2011). *The clean coder: A code of conduct for professional programmers.* New York, NY: Pearson Education.

Project Management Institute. (2013). *A guide to the Project Management body of knowledge (PMBOK® guide)* – Fifth edition. Newtown Square, PA: Author.

Schwaber, K. & Sutherland, J. (2016). *The Scrum guide™*. Available at

Originally published as part of the PMI® Global Congress 2016—North America proceedings.

http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf

Shalloway, A. (2009). *Lean-Agile software development*. New York, NY: Pearson Education.