



Lean Meets Agile Software Development & Project Management

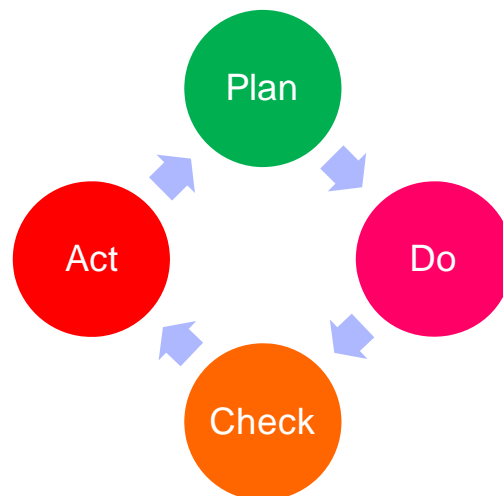
A WHITE PAPER PROVIDED TO ASPE BY RICHARD J PERRIN

Lean Meets Agile Software Development and Project Management

Richard J Perrin PMP CSM SSBB QFDGB

Abstract: This white paper discusses the roots of agile thinking in terms of the modern quality evolution which has been exemplified in the Toyota Production System (TPS). In viewing the TPS as the ultimate Lean implementation, we see great similarities to Toyota's 14 Points of the TPS (Toyota Production System), the Agile Manifesto and Agile Principles for software development, and will explore why agile approaches will become the primary method for developing software and managing projects within the next five years.

The Project Management Institute added a graphic near the beginning of the 2004 version of the PMBOK that bears mentioning here. It is a representation of the Shewhart-Deming Cycle originally developed by Dr. Walter Shewhart at Western Electric in the late 1930's. It basically looks like this:



This is called an empirical process because it demonstrates the idea that a theory can be verified or disproved by observation or experiment. In other words, we are looking at the fundamental scientific method.

Plan: Without a theory of reality, there is no plan – everything starts with a theory. Your plan becomes the vehicle for testing your theory.

Do: Based on the theory we do the work as defined in the plan – execute the plan accurately and with precision.

Check: We then check the result to see if what we thought would occur actually happened. What worked – what didn't?

Act: We hold the gains from the elements that worked. We make adjustments for the elements that didn't work and roll it, plus new elements, into the next plan.¹

Wash, rinse and repeat. The above cycle is the basis of incremental improvement for Toyota (TPS), and every manufacturer or service provider that produces high customer satisfaction products and services. It is also the fundamental basis of incremental or agile software development. Why is this true? Because the key element in the cycle is the notion of repetitive adjustment in the process or product, based on the feedback obtained from the customer.

Enter the Waterfall...

However since the 1970's a one-pass approach to building software took root and has remained the predominant approach for building software for the last 30 years or more. This model commonly referred to as the 'waterfall' approach, proceeds in a series of clearly defined steps that looks something like the following:

- Requirements
- Analysis
- High Level Design
- Detail Design
- Code
- Test
- Deploy

In 1970 Dr. Winston Royce, in writing about implementing large software projects, documented a software process that became known as the 'waterfall' process. Somehow – incorrectly – his article was misinterpreted by some who read it, and who subsequently applied it as the correct approach for developing software from that point on. Royce actually identified the 'waterfall' idea as an unworkable and 'broken' process. What he *did* say was that the 'waterfall' approach would only work if the correct *feedback* mechanisms were built into each phase to incrementally correct what was missed. What he basically said of the 'waterfall' approach was this: "...*the implementation described above is risky and invites failure*".²

The issues with this approach include some of the following:

- We assume that the entire process can be planned and estimated from the beginning even though the idea of 'uniqueness' indicates that we are either solving a new problem, doing something we have never done before or are dealing with uncertainty, unknowns or simply don't know what we don't know.
- The plan becomes etched in stone. It is change averse. Change requires a top heavy, time consuming and expensive process requiring: impact assessment, client review

and signoff and Change Control Board review. Change always increases cost, generally pushes out the timeline and usually occurs because:

1. A requirement was missed in the design phase
 2. User needs have changed due to market or competitive pressures
 3. Requirements were not clearly defined in the analysis phase
 4. Discovery: you were blind-sided by an event that no one predicted (or could have predicted)
- If there is a disagreement on the interpretation of the ‘requirements’, the business and IT dig in for a head-to-head confrontation. Change requires a Change Request (CR) The CR process is used by IT to control *Scope Creep*: a situation in which the business asks for a change to the requirements (from the IT perspective) but refuses to allot additional time or cost to implement the change because the business is claiming that this is what it meant in the first place. From the business’ perspective, IT becomes the bottleneck to implementing what the business has requested. From the IT perspective, the business is attempting to get something for nothing and hanging IT out to dry in the process...
 - The largest risks – integration of the major system components and migration to the production environment – occur at the end of the process. Addressing these risks late in the project can lead to explosive increases in cost, timeline or cause the project to fail outright³
 - If there are errors in the coded solution they only come to light in the final system Test phase – the most expensive place to correct the problem before the product is shipped to the customer. Fixing these errors may require changes to the Code, the Detail Design, the High Level Design the Requirements or all of these elements together.

When we are dealing with unknowns in a software project we have a theory of reality that at some point needs to be proven or disproved. In software development, the P-D-C-A cycle enables us to incrementally test reality, obtain feedback from the customer to make corrections and continually improve the product or process until the customer is ready to release the product for the business. The US quality gurus, Dr. W.E. Deming and Dr. Joseph Juran originally brought the concepts of incremental improvement to Japan in the early 1950’s. They not only promoted the idea of continuous, incremental improvement in the manufacturing environment but also promoted it for the business process as well.

Fundamental tools in the P-D-C-A cycle are:

1. The Pareto Chart (Finding the signal in the noise. – separating the ‘vital few’ from the ‘trivial many’ as Dr. Juran stated it. Identify the one or two issues/risks that will have the largest positive impact if you address them early in the process)
2. The Ishikawa Diagram (Cause-and-effect or root cause diagram)
3. The scatter diagram (Does changing ‘X’ have an effect on ‘Y’?)
4. Statistical process control charts (Identifying what variation in your process means. A key tool used to prevent management from tampering with the system and attempting to either 1) fix elements that are not broken or 2) ignoring broken elements that need to be fixed.)

Key tools in the Lean process include:

1. QFD – Quality Function Deployment. Developed by Yoji Akao in the late 1960's and implemented Mitsubishi's Kobe shipyards. QFD is a process that traces business needs to business requirements to technical requirements and captures the "Voice of the Customer". A key feature is that it focuses on fitness for use elements and explores how the customer actually uses the vendor's product or service.
2. AHP – Analytic Hierarchy Process. Created by world renowned mathematician Thomas L Saaty at Wharton in the early 1970's. The AHP is easily the most effective and tamperproof of the tools to help the customer prioritize business needs and requirements.

Agile Approaches

In the last 10 to 15 years, the use of agile software development approaches and agile Project Management approaches have delivered higher quality software, and higher customer value than their "waterfall" counterparts. If we examine some of the concept of agility, we find that the Agile Alliance developed the "Agile Manifesto" in 2001 in which the following values are described for developing high quality, high customer satisfaction software:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

While the elements on the right side of each bullet point are important, the Agile Alliance places more value on the elements on the left-hand side of each statement. It should be in no way interpreted that the Agile Alliance eschews processes, documentation, contracts or plans in the execution of a project or software process. What it does mean is that more value for the project and the customer are derived when we focus on interactions between individuals, delivering working software in increments, collaboration between the customer and the developer, and the ability to flexibly adapt to change without penalizing the customer or the developer because a change was necessary. This may seem like a radical new concept to many, however the origins of this thinking are deeply rooted in the work of W. Edwards Deming, Joseph Juran and subsequently the Toyota Production System.

Toyota created 14 points as a guideline for improving the quality and the effectiveness of a business⁴. These points and some selected subcategories appear below under the TPS column. Notice how the Agile Manifesto (AM) (www.agilemanifesto.org)⁵ and the Principles behind the Agile Manifesto (AP) generally map to the 14 points of the TPS. Some of the AM and Agile Principles may appear multiple times opposite the points in the TPS.

Toyota Production System (TPS)	Agile Principles (AP), Agile Manifesto(AM)
<p>1. Base your management decisions on a long-term philosophy, even at the expense of short-term financial goals</p> <ul style="list-style-type: none"> • Generate value for the customer, the society and the economy 	<p>AP: Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.</p>
<p>2. Create continuous process flow to bring problems to the surface</p> <ul style="list-style-type: none"> • Re-design work processes to achieve high value-added, continuous flow. Strive to cut back to zero the amount of time that any work project is sitting idle or waiting for someone to work on it. 	<p>Agile Manifesto: Working software over comprehensive documentation</p> <p>AP: Working software is the primary measure of progress.</p>
<p>3. Use "pull" systems to avoid overproduction.</p> <ul style="list-style-type: none"> • Provide your downline customers in the production process with what they want, when they want it, and in the amount they want. • Be responsive to the day by day shifts in customer demand rather than relying on computer schedules and systems to track wasteful inventory. 	<p>Agile Manifesto: Working software over comprehensive documentation</p> <p>AP: Continuous attention to technical excellence and good design enhances agility</p> <p>AP: Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.</p>
<p>4. Level out the workload.</p> <ul style="list-style-type: none"> • Eliminating waste is just one third of the equation for making lean successful. • Work to level out the workload of all 	<p>AP: Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.</p>

<p>manufacturing and service processes as an alternative to the stop/start approach of working on projects in batches that is typical of most companies.</p>	<p>AP: Simplicity--the art of maximizing the amount of work not done--is essential.</p> <p>AP: Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.</p>
<p>5. Build a culture of stopping to fix problems, to get quality right the first time.</p> <ul style="list-style-type: none"> • Quality for the customer drives your value proposition. • Build into your processes the capability of detecting problems and stopping itself. • Build into your culture the philosophy of stopping or slowing down to get quality right the first time to enhance productivity in the long run. 	<p>Agile Manifesto: Individuals and interactions over processes and tools</p> <p>AP: Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.</p> <p>AP: Continuous attention to technical excellence and good design enhances agility.</p>
<p>6. Standardized tasks are the foundation for continuous improvement and employee empowerment.</p> <ul style="list-style-type: none"> • Use stable, repeatable methods everywhere to maintain predictability, regular timing, and regular output of your processes. • Capture the cumulative learning about a process up to a point in time by standardizing today's best practices. Allow creativity and individual expression to 	<p>AP: The best architectures, requirements, and designs emerge from self-organizing teams</p>

<p>improve upon the standard; then incorporate into the new standards so that when a person moves on you can hand off the learning to the next person.</p>	
<p>7. Use visual control so no problems are hidden.</p> <ul style="list-style-type: none"> • Use simple visual indicators to help people determine immediately whether they are in a standard condition or deviating from it. • Design simple visual systems at the place where work is done, to support flow and pull. • Reduce your reports to one piece of paper whenever possible, even for your most important financial decisions. 	<p>Agile Manifesto: Responding to change over following a plan</p>
<p>8. Use only reliable, thoroughly tested technology that serves your people and processes.</p> <ul style="list-style-type: none"> • Use technology to support people, not replace people. Often it is best to work out a process manually before adding technology to support the process. • Reject or modify technologies that conflict with your culture or that might disrupt stability, reliability, and predictability. • Nevertheless, encourage your people to consider new technologies when looking into new approaches to work. 	<p>AP: Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.</p>
<p>9. Grow leaders who thoroughly understand the work, live the philosophy, and teach it to others.</p>	<p>Agile Manifesto: Customer collaboration over contract</p>

<ul style="list-style-type: none"> • Do not view the leader's job is simply accomplishing tasks and having good people skills. Leaders must be role models of the company's philosophy and way of doing business. • A good leader must understand the daily work in great detail so he or she can be the best teacher of your company's philosophy. 	<p>negotiation</p> <p>AP: Business people and developers must work together daily throughout the project.</p>
<p>10. Develop exceptional people and teams to follow your company's philosophy.</p> <ul style="list-style-type: none"> • Train exceptional individuals and teams to work within the corporate philosophy to achieve exceptional results. Worked very hard to reinforce the culture continually • Use cross functional teams to improve quality and productivity and enhance flow by solving difficult technical problems. 	<p>AP: Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.</p>
<p>11. Respect your extended network of partners and suppliers by challenging them and helping them improve.</p> <ul style="list-style-type: none"> • Challenge your outside business partners to grow and develop. It shows you value them. Set challenging targets and assist your partners in achieving them. 	<p>AP: Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.</p>
<p>12. Go and see for yourself to thoroughly understand the situation.</p> <ul style="list-style-type: none"> • Solve problems and improve processes by going to the source and personally 	<p>AP: The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.</p>

<p>observing and verifying data rather than theorizing on how the basis of what other people or the computer screen tell you.</p>	
<p>13. Make decisions slowly by consensus, thoroughly considering all options; implement decisions rapidly.</p> <ul style="list-style-type: none"> • Do not take a single direction and go down that one path until you have thoroughly considered alternatives 	<p>AP: Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.</p>
<p>14. Become a learning organization through relentless reflection and continuous improvement.</p> <ul style="list-style-type: none"> • Once you have established a stable process, use continuous improvement tools to determine the root cause of inefficiencies and apply effective countermeasures. • Design processes that require almost no inventory. This will make waste of time and resources visible for all to see. 	<p>AP: At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.</p> <p>AP: Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.</p>

There are several key elements that need to be emphasized here. Toyota's TPS and the Principles of Agile are fundamentally agreed on the following;

1. Building quality into the product or service instead of inspecting it into the product at the end of the development cycle.
2. Focus on continuous improvement.
3. Face-to-face communications are more important than e-mails or what the user sees on a computer screen.
4. Engage highly motivated individuals to get the job done i.e. get 'the right people on the bus'.
5. Reflection is a key element in helping the team to continuously evaluate and improve its effectiveness.
6. Live and breath the philosophy of improvement and be an example to your employees, your customers and your vendors

The growth of agile processes for both software development and software project management is a natural evolution of the lean processes developed in manufacturing in the 1980s. However the issues specific to software development are somewhat unique. The DNA of software development and the infrastructure on which it exists are currently changing at a pace that is unequalled in the history of *any* technology. Not only has Moore's law proved to be correct – CPU capacity approximately doubles every 18-24 months while the cost is reduced by 50%⁶ - but the evolution of the hard-drive has accelerated at an even more rapid rate. In 1969 IBM implemented changes to its mainframe O/S averaging once every 2-3 years. Today, modern software implementations are updated on a quarterly basis or sooner.

Given that changing markets, changing business needs, new requirements, and paradigm shifts forces the business to adjust its processes to a changing environment, the monolithic project process actually constrains the organization with an outmoded project model that is change averse, top-heavy, and penalizes both the developer and the user for "inaccurate" estimates. After all, how is it possible that after 10 years of disciplined project management implementation between 1994 and 2004 that 65% of the 40,000 projects assessed by the Standish Group in the Fortune 500 are either challenged or fail? It could be that most of the Fortune 500 is using the wrong model for developing software, and that model is the "waterfall" model.

company – a world class exemplar of the waterfall process - to produce noticeable improvements in delivery time and defect reduction utilizing SCRUM: <http://jeffsutherland.com/scrum/Sutherland-ScrumCMMI6pages.pdf>. Also notice how SCRUM processes coincide with many CMMI level 5, key processes.

Is IT a Commodity?

Implementing disciplined, agile approaches for software development may be the key element in enabling US software companies and companies implementing software projects to stay competitive in the global marketplace for the foreseeable future. While IT ‘pundits’ such as Nicholas Carr have frequently written off IT as a ‘commodity’⁷ (the idea that building and maintaining an IT infrastructure does not give any business a competitive advantage if everyone has it), the industry tenaciously continues to defy this perception/wish/delusion in very practical terms. This is simply because business users are as unique as the software products they request from the software development team. It is doubtful whether IT will ever actually become a commodity unless business users themselves view their own work as a commodity for which it is traded or sold like frozen concentrated orange juice or pork bellies. I’ve never met a business user that thinks of their work in this way.

With the implementation of agile processes for software, the US software industry is close to achieving what was achieved for manufacturing in the United States in the early 1980s, namely the ability to build quality into the product instead of trying to inspect it into the product, deliver user value consistently, deliver quickly (customers *love* speed), deliver accurate cost and timeline estimates, and ultimately give management unprecedented visibility into the software process. My prediction is that within five years, agile development and agile project management approaches will be required for any business that delivers or develops software to remain competitive, or risk the potential of going out of business. Is IT a commodity? Arguably, it is not. Delivering value to the customer with speed and quality creates a competitive advantage for any business. Just ask Toyota.

¹ *Statistical Method from the Viewpoint of Quality Control*, Dr. Walter Shewhart, 1939

² . “*Proceedings, IEEE WESCON, August 1970, pages 1-9*. Copyright © 1970 Institute of Electrical and Electronics Engineers. “

³ *Agile and Iterative Development*, Craig Larman

⁴ *The Toyota Way*, Jeffrey K Liker, 2004 McGraw Hill

⁵ <http://www.agilemanifesto.org>

⁶ *Cramming More Components onto Integrated Circuits*, Gordon E Moore, Electronics, Volume 38, Number 8, April 19, 1965

⁷ *Does IT Matter?* Nicholas G. Carr, Harvard Business School Press; 1 edition (April 2004)