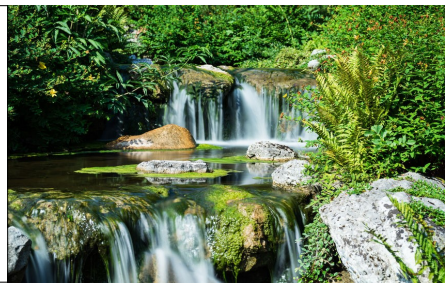
Agile, Waterfall, and Lean

Emerging Trends in Project Management

Steve Wille

Project Phases

1. Initiating
2. Planning
3. Executing
4. Controlling
5. Closing

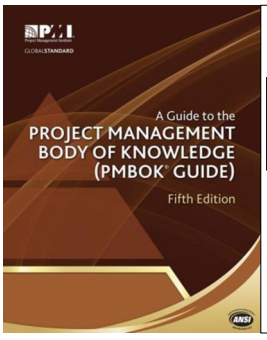


Business moves fast and is accelerating. Management techniques that were fast enough in the nineteen-hundreds can be too slow in a competitive world where someone somewhere is inventing the next new thing. All areas of the business can learn techniques from the software development world where speed counts, and agility in meeting customer needs is critical. Early in the 21st century, a group of software developers published the Agile Manifesto. It slowly caught on and over the past several years agile software development has rapidly hit mainstream. It is highly likely that every area of every business is going to need this trend in agility to survive.

Traditional **waterfall** refers to a sequential software development processes, in which progress is seen as flowing steadily through the project phases. The water flows in one direction symbolizing that once a phase is completed you don't get to go back to make changes. This is because changes are expensive late in a project time line as compared to getting things right in an early phase. Overall project efficiency is the goal of waterfall development. Our source for defining traditional waterfall project management is the Project Management Institute's book, *A Guide to the Project Management Body of Knowledge (PMBOK Guide*). This was first published in the 1990s. It documents good practices in project management. The traditional phases of a project are *initiating, planning, executing, controlling,* and *closing*.

PMBOK covers ten knowledge areas with particular attention given to **scope**, **time**, and **cost**. These are known as the triple constraints. It is easier to accomplish two of the triple constraints if you don't care about the third. For example, it is easier to be on time and on budget if you don't care about scope as compared to being on time, on budget, and with full scope.

In all PMOBK knowledge areas ***management*** is a key word. There are many definitions of management. We will define it as the opposite of unmanaged, chaotic, and out of control.



Integration Management
Scope Management
Time Management
Cost Management
Quality Management
Human Resource Management
Communications Management
Risk Management
Procurement Management
Stakeholder Management

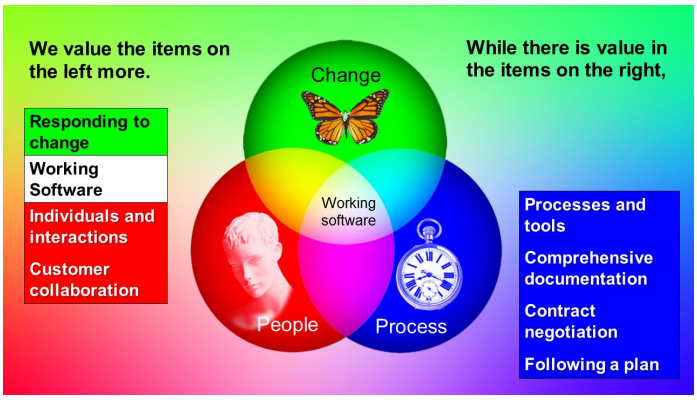
Agile projects are like rafting down the rapids. Change can happen any time, even late in the process. Watch where you are going and think fast!

The dictionary definition of agile is, "marked by ready ability to move with quick easy grace, *an agile dancer*. " The second definition is "having a quick resourceful and adaptable character, *an agile mind*" (merriam-webster.com) . The theme here is flexibility and speed, with resourcefulness and adaptability.

An interesting book on this subject is, *Age of Speed, Learning to Thrive in a More-Faster-Now World,* by Vince Posente. It's a simple issue of supply and demand. There is an increased demand for time, but a virtually static supply of it. The solution to this is speed. Posente suggested that as we gain wealth and want to do more, time becomes a limiting factor. To do more we have to do everything faster, squeezing more into the same amount of time.

Few businesses are more traditional than the London black cab. In the days before Google Maps, the most efficient way to find a building location in London was to jump into a black cab. The drivers had to pass a test to prove they could find their way around this ancient city with streets that go every which way. Drivers could not learn this skill overnight. Often it would take several years of study and practice to pass the test. On average it took twelve times to actually pass the test. In 2015 the cabby school, Knowledge Point, shut its doors. GPS and Uber cab changed the game. Uber is more agile. Agility wins.

Consider the life of a fruit fly that has just 10 to 18 days to live. Even if conditions are prefect for them their maximum life is under two months. If they want to get anything done, they had better do it quickly, starting with the need to mate and reproduce. Geneticists like studying fruit flies because they can watch multiple generations of these creatures adapt to their environment. Clayton Christensen, in is book, *The Innovator's Dilemma, When New Technologies Cause Great Firms to Fail,* saw disk drive manufacturers as his fruit flies to study. Each time the outside environment changed in disk drive demand, the dominant manufacturer was displaced by an upstart that paved the way for smaller, less expensive



disk drives. Christensen identifies common reasons the leading suppliers did not make it to the next generation.

Software development is the fruit fly of the business world. We are always chasing the next new thing. Yesterday's solutions no longer apply to today's opportunities. People who have spent their careers in the world of computer application development have had to learn, unlearn, and relearn everything they knew about software development every time they started a new project.

We will define agile project management as it is expressed in the *Agile Manifesto,* published in 2001. This is a brief document containing four high level strategies and twelve principles. You can find this on line and print it yourself.

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

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

That is, while there is value in the items on the right, we value the items on the left more.

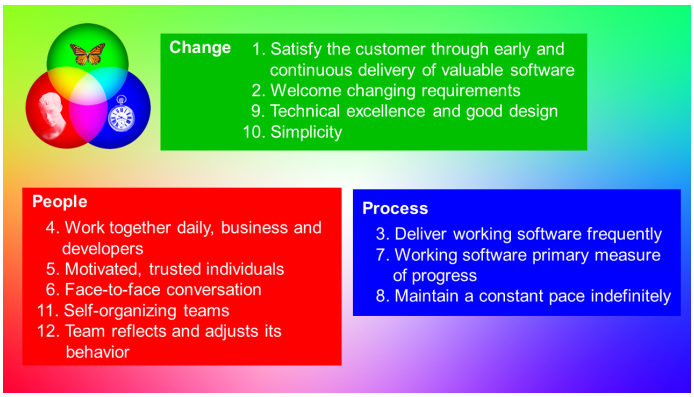
We follow these principles:

1. Our highest priority is to **satisfy the customer** through **early and continuous delivery** of valuable software.
2. Welcome **changing requirements, even late in development**. Agile processes harness change for the customer's competitive advantage.
3. **Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must **work together daily** throughout the project.
5. Build projects around **motivated individuals**. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation**.
7. Working software is the primary **measure of progress**.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to **maintain a constant pace indefinitely**.
9. Continuous attention to **technical excellence and good design** enhances agility.
10. **Simplicity**--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from **self-organizing teams**.
12. At regular intervals, the **team reflects** on how to become more effective, then tunes and adjusts its behavior accordingly.

As we examine the manifesto, it is useful to break it into parts. We will use the *Colorful Leadership* model where red, green, and blue light come together to make white light. Each color represents something different, and only when all three come together in equal intensity do you get what you are really looking for. We use 3 colors rather than two so we do not fall into a polarizing trap where it is either one thing or the other. The manifesto values the things they place on the right, but they value the left more. The *Colorful Leadership* model breaks the things on the left into two parts, making these two parts equal in importance to the one color on the right.

We represent items on the right with blue light. The manifesto focuses on four items: processes and tools, comprehensive documentation, contract negotiation, and following a plan. With red, we focus on things that are people oriented, specifically, individuals and interactions and customer collaboration. With green we represent change. This model suggests that process, people, and adapting to change are equally important. If one color is left, the overall effect is off-color, meaning we are less effective than we could be with all three.

The value of the model identifying three equal components is you can examine each individually on its own merits. Process is what managers manage. The goal is to build a well oiled machine where a repeatable process



is followed by everyone to achieve predictable results. Without a strong focus on process a business is not sustainable over time. With a process focus, efficiencies can be achieved. With the process defined you can send people to training to understand the theory and practice process skills.

People are a bit more challenging than processes. You cannot manage people, instead, you need to lead people They are living, independent, social beings who come to work with their own attitudes and capabilities. The role of a leader is to set up an environment where people might want to do what the leader wants them do.

Change is represented by the green light in this model. The butterfly effect is very much at play meaning small changes can make big differences. We can never predict exactly what will happen in a complex system.

People Oriented Agile Principles

4. business people and developers must work together daily throughout the project.

Discover magazine, in the March 2014 issue, reported that at Bank of America call centers, some teams were more productive than others. Call centers have extensive records to measure efficiency, and they have a large enough population of workers to see meaningful patterns regarding worker efficiency. At first, they looked to see if there were differences in the people among the different work groups, but they found no meaningful differences in qualifications, experience, or training. Failing to find an obvious reason for the differences in team productivity, they watched interaction patterns and discovered that teams that took breaks together were 15 to 20 percent more productive than those with staggered breaks. It was not known what people talked about on their breaks, just that they took break time together.

It seems to be a human trait that when people spend time together, they are more productive than if they were apart. By creating one team of business people and developers there is a predicable gain in effectives.

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

Stephen M. R. Covey, in his book, *The Speed of Trust*, noted that when people show a high degree of trust they work faster. There is no need to cover their back sides with unnecessary documentation, just in case something goes wrong. Trust is not something you give. It is something that happens as people work together. Over time, we learn when we can trust and when we must verify.

6. the most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

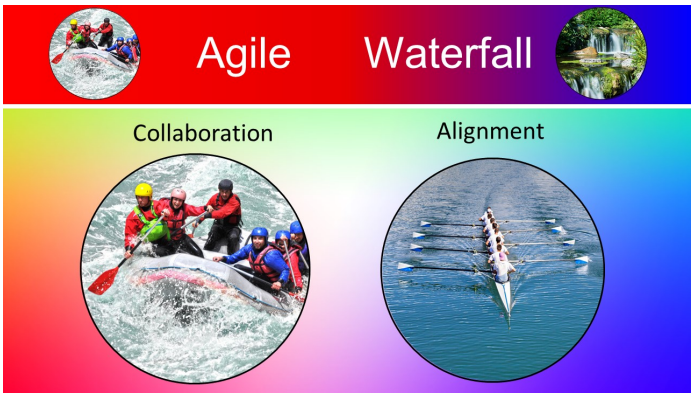
Wayne Baker in his book, *Achieving Success Through Social Capital,* notes that simple human interaction builds social capital, much like economic capital. When you deposit money in the bank, you can draw from it when you need it. Building social capital gives you connections to people you can draw on when you need them. Baker points that for a human connection nothing competes with face-to-face conversation.

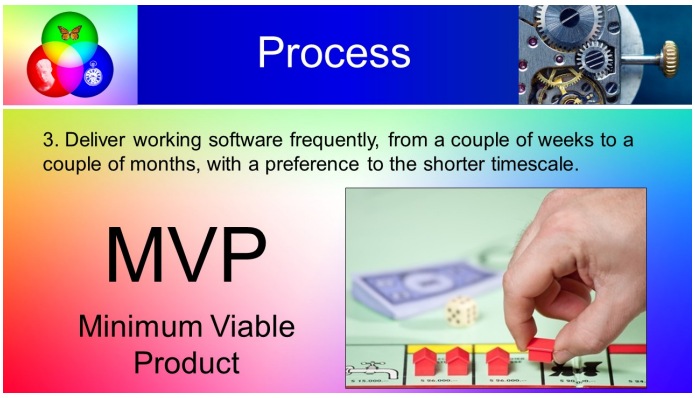
11. The best architectures, requirements, and designs emerge from self-organizing teams.

Rod Collins, author of, *Wiki Management*, notes that nobody is faster than everybody and nobody is smarter than everybody. He suggest we get away from 19th century agricultural and 20th century industrial models for organizing work teams. We no longer need a hierarchy to tell us how do our jobs. We are interconnected globally with smart people everywhere. If you want people to be fully engaged in a project, you need to let them get engaged and then stand back to encourage and support them. In a private conversation with the writer of this presentation, Collins noted that empowerment is hierarchical idea where authority is delegated from on high. It was an effective management and leadership tool in the 20th century. Today, he recommends thinking in terms of enablement rather than empowerment. Give people the tools and support to enable them to do the work, but let the work teams self organize as it works for each team.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Periodic reflection is fundamentally different from the final step in waterfall when there is a lessons learned meeting. At that point, not only is it too late to change anything, but the focus tends to be on techniques that





worked or did not work. This goes into the project documentation and is promptly buried because it is of little immediate value. In the agile world, the periodic reflection is known as a retrospective. The team owns it and there are no formal, public meeting minutes published. It is an opportunity for the team to reflect on what is working and should continue, plus what is not working and should change or be stopped.

You do not need to following the agile method to do periodic retrospectives. You simply need to do them at regular intervals and let the teams own them.

Clear roles - Ambiguous roles

Sometimes we need clear roles and other times we need ambiguity with self-organizing teams. It is great to identify who is responsible for what, and it is also great to give people freedom rather than putting them in a predefined box. If you want innovation, you need ambiguity. If you want predictability and control you need clarity. Can you do both on one project? Of course you can. Some tasks need great clarity and other tasks require creativity. Too much control kills creativity.

The agile principles that address this are:

- 4. Business people and developers must **work together daily** throughout the project.
- 6. The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation**.
- 11. The best architectures, requirements, and designs emerge from **self-organizing teams**.

By contrast, traditional waterfall projects usually clarify roles, often through the *RACI Matrix*:

- **R**esponsible (Persons who will be performing the work)
- **A**ccountable (Person who is ultimately held accountable)
- **C**onsulted (Subject matter experts)
- **I**nformed (Communication is one-direction)

Alignment - Collaboration

Consider a rowing team on relatively flat water. The rowing team is most efficient if everyone is aligned, doing the exact same thing at the exact same time. To accomplish this, in the crew there is a coxswain who sits in the back facing the front. The coxswain is responsible for steering the boat, and coordinating the power and rhythm of the rowers who are facing the back and cannot see where they are going.

Next, consider a rubber raft running the rapids in fast moving, dangerous water. People must coordinate their efforts, but due to rapidly changing conditions they must be flexible. Everyone faces the front and pays attention to where the raft is going. There is no coxswain keeping time. Instead, the leader is yelling out what needs to be done. Can one athlete do both? Of course, but not at the same time. When rowing, alignment counts. When paddling, experienced judgment counts. Some aspects of a project require an aligned team and other aspects require experienced judgment with a great deal of flexibility. The mistake is to be a coxswain when a different kind of leadership is needed.

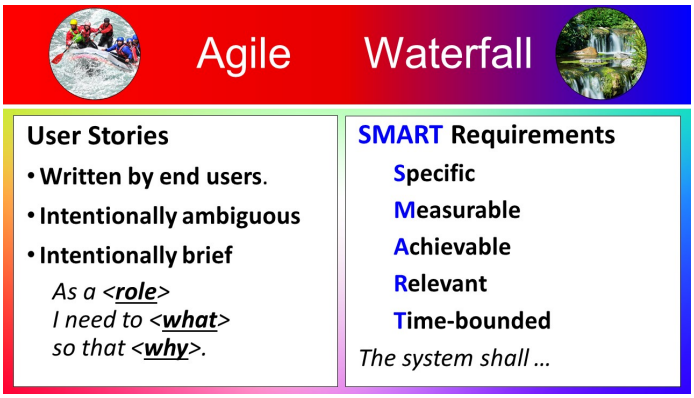
Process Oriented Agile Principles

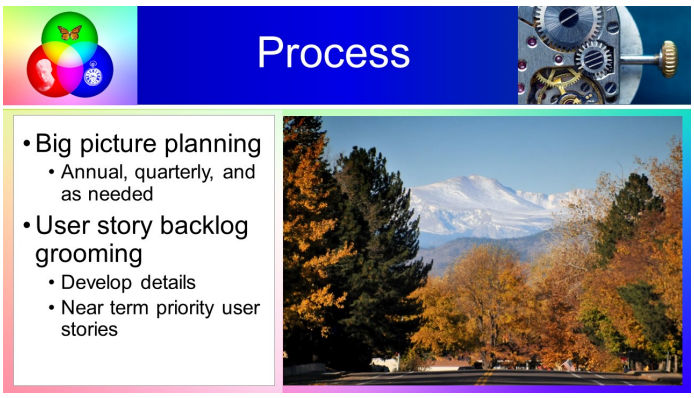
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Frequent delivery of something that works is at the core of the agile method. You don't have to do it all at once. What you need is a *minimum viable product.* This is something that works when it is delivered and more can be added later. Think of the Monopoly game. First you buy the properties, then you build houses, as time goes by, when you can afford it, you build hotels. Is it possible to break your business project into a series of short term deliverables that will add value? All the while, everything you deliver works.

7. Working software is the primary measure of progress.

We like measurements. In school we want all As. In project management we often measure against plan. How are we doing compared to the estimate? Are we on schedule? We want the numbers to be good. If the measurements are not related to results, we can get good scores, even with bad results. Agile looks to the team to set its own measurements and compete with itself. User story points mean nothing to the outside world. They are not like hours which are specific and measurable.





Instead, the team looks at a user story and gives it a relative score for effort, complexity, and doubt. The goal is to understand how much can be delivered by the team in a set period of time. Over time, the team may become more efficient and deliver more story points within the same amount of time.

Requirements - User stories

Project scope defines what needs to be accomplished. In the waterfall world, requirements define the scope. ***SMART*** requirements are ***S****pecific*, ***M****easurable* ***A****chievable*, ***R****elevant*, and ***T****ime bound*. There is to be no deviation from these specific requirements without formal approval, even if you could do something better, faster, and at lower cost. You may not deviate without approval.

In the agile world there is intentional ambiguity in defining what needs to be done. User stories are commonly used and they are written from the user's perspective. They define what is needed and why it is needed. The agile team is expected to focus on ***why*** when viewing the user story and figure out how to best accomplish this objective. There is room for creativity. Ideas on accomplishing the goal better, faster, and at lower cost are always welcome.

User stories and requirements can both be used for quality assurance testing. Requirements are specific enough to be inserted directly into the test plan. Acceptance criteria are frequently included in the user story and also can be inserted into the test plan. Here are some examples:

Requirements

1. The system shall interact on-line with Mastercard, Visa, American Express, and Discover cards.
2. The system shall transmit name, address, phone, purchase amount, credit card number, and expiration date to the credit card company.
3. The system shall record the authorization from the credit card company.
4. The system shall comply with PCI security requirements for credit cards.

User Story

As a sales agent, I need to accept credit cards over the phone and verify the information so that I can complete the purchase.

Acceptance Criteria

1. This works with all credit cards accepted by our company.
2. There is a way to verify the information with the credit card organization.
3. Customer data is kept secure.

The *International Institute of Business Analysis (IIBA)* has recognized the evolving face of how requirements can be written. The current version of the BA-BOK says, "The nature of the representation may be a document (or set of documents), but ***can vary widely*** depending on the circumstances." Clearly you can do both, depending on what is needed for the project. If you know exactly what you want, go for requirements. If you are interested in what you think you want, or something better, then go for user stories. There is no reason not to use both on the same project, depending on the specific needs to be accomplished.

Complete details - Big picture strategy

Agile and waterfall projects both require planning, but they require different types of planning. With waterfall there is detailed planning followed by signoff. The detailed plan is the primary monitoring and controlling mechanism. If it is in the plan it must happen and if it is not in the plan it must not happen. Agile projects have a general big picture plan, and a bunch of user stories. The user stories are on the wall or in a database, but they remain un-prioritized until it is time to prioritize them for the next work effort. At that point the top priority user stories are built out into greater detail and the work gets done.

Agile teams include the product owners who have great influence on the priority. The work team also helps sets the priority because some things are best accomplished before other things. There is a bit of chaos in the process, but from chaos, order can emerge.

Waterfall people might question the value of chaos and wonder if it can work. The big picture plan defines the overall project and the focus remains on accomplishing that objective.

Change

If we were manufacturing cookie cutter products in a factory, we would not expect much change, other than relatively minor changes resulting from continuous improvement in the manufacturing process. Software development is the opposite of this. Change happens because development is a creative process, and as a product is developed, more is learned about it. The product owner will see new possibilities. In addition, as the external market changes, along with enabling technology. If we don't accept change, we get left behind.

We follow these principles:

1. Our highest priority is to **satisfy the customer** through **early and continuous delivery** of valuable software.
2. Welcome **changing requirements, even late in development**. Agile processes harness change for the customer's competitive advantage.

The number one agile principle is to deliver early and often so changes can be explored and incorporated into the next iteration. Thus, agile teams deliver new releases frequently.

Timing: Random—Sequential

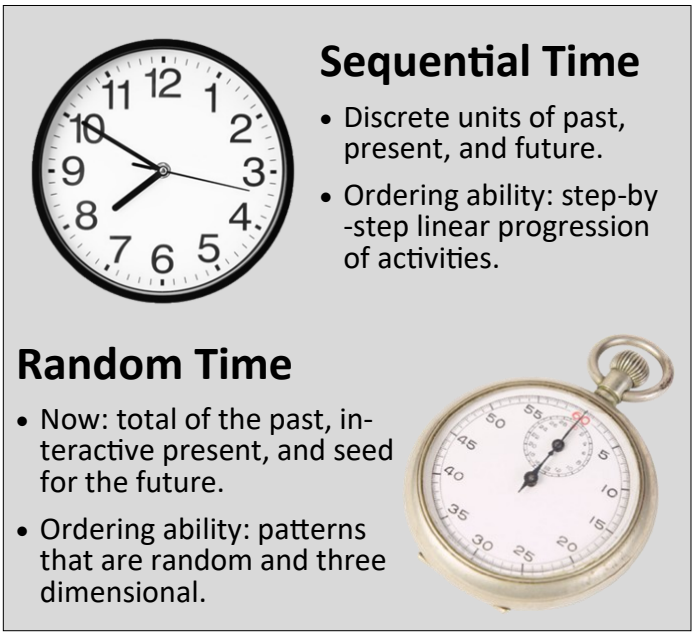
Timing is everything. Ask any musician and you will learn that timing makes the music come alive. The same is true with projects. One of the fundamental differences between agile and waterfall projects is timing. It is also one of the areas driving the greatest division of opinion. What is the right timing for a project task? Waterfall calls for sequential, step-by-step timing with an emphasis on the critical path which is identified during the planning phase. Changes require formal approval because a change may affect the project cost and schedule.

Agile promotes more random timing with a focus on the next deliverable, along with creating a minimum viable product. Changes to requirements, even late in development are welcome.

Opinions on the best timing often reflect personal preferences rather than project needs. Anthony Gregorc noticed difference in how we perceive time and saw preferences in how we handle timing. In *An Adult's Guide to Style* he identified predictable patterns. He took no stand on what is best, just that there are common differences in style.

We can separate people into groups. Some people lean toward what Gragorc calls ***random*** and some toward ***sequential***. This is a continuum, with relative few at either extreme, but the extremes are useful for defining the continuum.

People who lean toward sequential timing see time in terms of discrete units, such as minutes, hours, days, and years. Time always moves forward from past to present to future. The clock never stops and never goes backwards. This pattern calls for step-by-step tasks that lead directly to the completion of a goal. People at the extreme of sequential are quite bothered when something is done in the wrong order. To them it destroys the whole process. They think you should take your time on each task to do it right the first time.



On the other side of the continuum there are people who see time as ***now***. Our graphic shows a stopwatch. If something does not work, stop the clock and try again. Doing it right the first time makes no sense. Until they have tried several times, they don't know what is ***right***. Rather than a linear progress of activities, the random style allows work in several dimensions at once. This may look random to a linear thinker, but to the random style person it is simply living in the ***now*** timeframe and working in several dimensions at once.

After many years of managing project teams I have seen more conflict resulting from differences in of timing than any other source. It is one of the fundamental reasons some people have a difficult time in either an agile or waterfall environment. One day I got a call from my son who was in his final semester of college. He said he was struggling with a required class. The class was analysis and design, which was part of his information systems major. I asked what the problem was. He said the professor wanted him to write the design before writing the software code. I told him that half of the experienced software developers I have known have the same problem. Because they fit the random profile I have told them to write the code first, but don't tell anyone. After that, write the design, which is just a description of the code they wrote. The sequential thinkers cannot handle things done out of order. I suggested to may son that he write the code, but not tell the professor. Give her the design before showing the program. At a reasonable time later, present the program. Then, tell her how wonderful it was to have a design. Be silent on when the design was written.

People who live in ***now*** need to maximize ***now***, and that means jumping right in. They have no problem

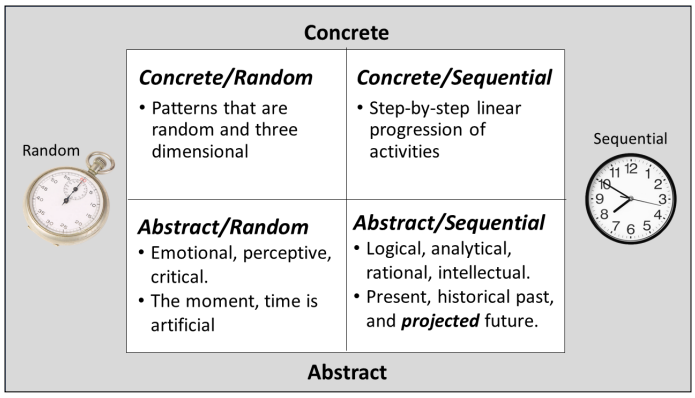
stopping the clock and starting over. An excellent software developer on one of my teams, Dave, often went through many iterations, starting over with different approaches until something worked. I respected Dave's approach, but when he worked on another development team that was much more structured and sequential, Dave was not respected and not well liked. Dave was my go-to person but he simply could not function in the other team's sequential environment. It was interesting that the other team was using the agile methodology, but they were anything but flexible, making them less than agile.

How do you do both? A better question would be how do you accommodate both styles? Chances are, you have a significant number of both styles on your team. I lean toward the random side and I tend to frustrate the sequential style people. They work real hard to get each step exactly right, and then I do a restart which invalidates the work they have done. I learned that when I am working with a sequential worker, I need to do my homework and think through exactly what I want. When I work with the random workers, it is best if I speak of the end goal in general terms and leave them alone. From time to time I check in to see if they are delivering approximately what I want.

If you are a hard core agile project manager, you will probably frustrate the people who have a need to work sequentially. If you are a hard core waterfall project manager you will probably frustrate the people who have a need to be more random. You must do both if you want everyone to be fully engaged and working to their full potential. If you do one or the other, exclusively, you will be half as effective overall. It is like playing rock, paper, scissors and always being a rock or always being a scissors. This is the subtractive zero sum process rather than the additive non-zero sum process.

With our first look at Gragorc's definition of the random and sequential styles, we were focusing on how work gets done. Gregorc called this the *concrete/random* and *concrete/sequential* styles. He also looked at this continuum from an abstract, non-physical standpoint. How do people think, and how do people feel? This he called *abstract/random* and *abstract/sequential*. The *abstract/sequential style* people prefer logic, analysis, knowledge, facts, and documentation. In the *abstract/random* style people think in emotions, relationships, and memories. The highest priority for an agile team is to satisfy the customer. This is an emotional objective. In contrast, the waterfall effort at stakeholder management is a logical and rational objective with a focus on knowledge, facts and documentation. Get stakeholders to sign a document on expectations so if they are eventually unhappy, you can prove that they agreed to that which now makes them unhappy.

User stories are preferred by the *abstract/random* style because of the customer focus. User stories allow room for emotion and happiness due to the flexibility and explanation on why something is needed. Traditional requirements fit the needs of the *abstract/sequential*



style. They focus on logic, facts and documentation, taking the emotions out.

Gregorc can be seen as a four-quadrant style profile, drawn here with the ***concrete-abstract continuum***, on the vertical axis and the ***random-sequential continuum*** on the horizontal axis. This should not be confused with other four-quadrant models that look at different behavioral characteristics. The thing to learn from Gregorc's model is that it is predictable that people will have preferences for the project methods that meet the needs of their styles. Some people will never be comfortable with methods that don't meet their needs. Whether you go waterfall or agile, some people are going to have to compromise.

Concrete/Random

- Instinctive, intuitive, impulsive, independent.
- Now: total of the past, interactive present, and seed for the future.

Concrete/Sequential

- Instinctive, methodical, deliberate, structured.
- Discrete units of past, present, and future.

Abstract/Random

- Emotional, perceptive, critical.
- The moment, time is artificial and restrictive.

Abstract/Sequential

- Logical, analytical, rational, intellectual.
- Present, historical past, and projected future.

Sequential team - Random team

When we talk about teams it is useful to relate teamwork to sports teams. The problem with this is there are many types of sports teams and many types of teamwork. Consider a baseball team. At any one time, one person has the ball and the people on the rest of the team are standing in their various positions, watching and waiting. The other team has one person at bat, and the rest are sitting on the bench watching and waiting. Baseball is a slow game, but a very precise game with no room for error.

Consider a basketball team. One person has the

ball, but the ball can be thrown to anyone who is available. Anyone can shoot for the basket at any time. Basketball is fast paced with plenty of opportunity for high scores. Speed and agility are the key to success.

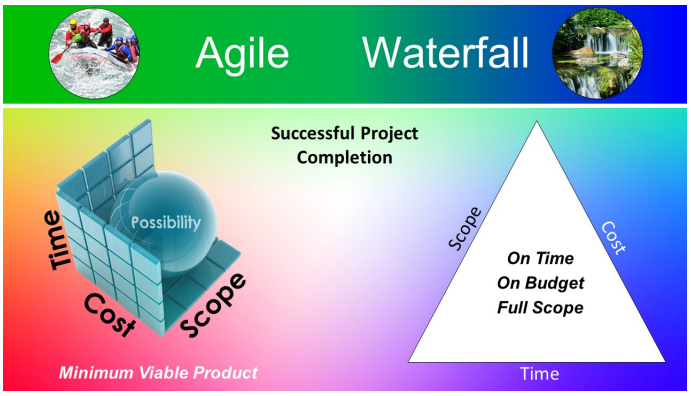
A sequential team at work is like a baseball team. It can have workers in cubicles waiting for someone to throw work over the wall. It is essential that each team member do his or her job error free because each person does a specialized part of the total work. The project manager spends a great deal of time sequencing and scheduling work to assure the right resources are available at the right time. People do their assigned work and nothing more. They never step out of their respective roles. It is not important for them to like each other, nor do they need to interact face-to-face, as long as the work gets done. The sequential team is well suited for waterfall projects.

A more random team at work is like a basketball team. Team members need to interact directly, preferably in the same space at the same time. The open office layout promotes this. Roles and responsibilities may be defined, but opportunities may arise for people to step out of their roles and do what needs to be done. The random team is well suited for agile projects.

Good Design and Simplicity

9. Continuous attention to **technical excellence and good design** enhances agility.
10. **Simplicity**--the art of maximizing the amount of work not done--is essential.

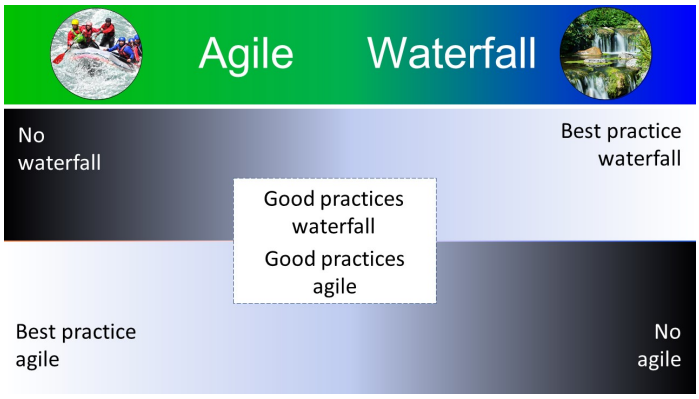
Because change is expected, it essential do design software to be maintainable and modifiable. If the system is highly complex in design, future changes will be difficult. An effective measure of good design and simplicity is to have a peer review where reviewers are expected to figure out how the system works and identify where changes can be safely inserted. An agile quality standard for software is it must include changeability. The organization wants to remain agile to survive.



Time, Cost, Scope

The goal of waterfall is to meet all three triple constraints. Achieving that is difficult. Consequently a large number of projects fail to be on time, on budget, with full scope. Agile takes a more realistic view by prioritizing scope rather than promising it all at once. Typically time and cost are locked in and the team delivers what it can within these constraints. It is important to deliver a minimum viable product early in the time schedule so that the customer has something working in production, even if the rest cannot be completed. Once the minimum viable product is in place, the product owners can decide when to end the project. If they are satisfied with progress, they might keep the project going by giving it more time and more budget to get more scope.

It is decision time. What are you going to do? The zero-sum way is to follow best practices for your preferred method. The non-zero sum way is to learn from a negotiation method where you try to figure out what each party really wants, and what each party will minimally accept. This defines the range of acceptability, which is the area up for negotiation. It is also known as the area where there is a win/win solution. If you take a blended approach, you will not be following *best practices* for either approach. But you can follow *good practices* from each.



If you know what you want and it has been done before, then creativity and innovation are not needed or wanted. Best practice waterfall might be the best choice. On the other hand, if it is entirely new, innovation and creativity are essential so best practice agile might be the be the best choice. Most projects are somewhere in the middle.

Lean

Lean comes from the manufacturing world. Agile comes from the software development world, Often the two terms are used in the same phrase because useful ideas from one domain can be helpful in a totally different domain.

Historical Background

Toyota is credited for developing what has become known as Lean manufacturing. In the early 20th century, Sakichi Toyoda, in a textile factory had looms that stopped themselves when a thread broke. This reduced waste. The modern Toyota Production System focuses on the systematic elimination of waste and increasing value to the customer. Lean practices, like *just-in-time* delivery of supplies, have been imitated world wide in manufacturing because they increased overall efficiency.

The difference between manufacturing and software development is how the triple constraints apply. In manufacturing, time, cost, and scope are known for the products coming off the line, and the goal is to reduce time and cost while making the product better, but this is done over time through a process known as continuous improvement. Software development, on the other hand, has massive unknowns as a project is started, and not until the project is completed are the triple constraints known.

Surrounding the environment of software development is a process of developing software. The process can be improved using certain Lean principles. Thus, we have popular methods such as Scrum, Kanban and Lean Agile. It is beyond the scope of this paper to reach into deeply into these methodologies because the focus on the Agile Manifesto that sparked new methods for significantly improving the software development process.

The thing to keep in mind is that manufacturing of a known product is different from designing and building an unknown product. Geoff Nicholson, inventor of 3M PostIt Notes® noted this fundamental difference when he said, "The Six Sigma process killed innovation at 3M. Initially what would happen in 3M with Six Sigma people, they would say they need a five-year business plan for [a new idea]. Come on, we don't know yet because we don't know how it works, we don't know how many customers [will take it up], we haven't taken it out to the customer yet."

This takes us back to good practices rather than being slaves to best practices. Just as you can adopt good practices from both agile and waterfall, you can apply good practices from Lean in the agile software development. You can also adopt agile good practices in Lean manufacturing world.

Note:

This paper is a compilation of other white papers on my web site. The intent of the paper is to define the choices we now see in project management methods with the purpose of understanding the differences so as individuals we can be effective in any environment. Often we don't get to chose the methodology so we have to live with it. Even so, we can use good practices from any method when it is appropriate. The key is to understand the purpose behind the methodology and use it effectively, rather than being a slave to the process.

Sources:

Gregorc, Anthony (1982), *An Adult's Guide to Style*, Columbia, CT: Gregorc Associates, Inc.

Agile definition: https://www.merriam-webster.com/dictionary/agile

Agile Manifesto http://agilemanifesto.org/ © 2001, the above authors this declaration may be freely copied in any form, but only in its entirety through this notice.

Christensen, C. (2013). *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail.* Cambridge, MA: Harvard Business Review Press.

Collins, R. (2013). *Wiki Management: A Revolutionary New Model for a Rapidly Changing and Collaborative World*. New York: AMACOM American Management Association.

Covey, S. M. R. (2006). *The Speed of trust: The one thing that changes everything.* New York, Simon and Schuster.

Creneti, J. (2008). *Insects & bugs: How long do fruit flies live?* Retrieved from https://www.youtube.com/watch?v=JiP3sg2XYu8

Coxswain definition: https://en.wikipedia.org/wiki/Coxswain_(rowing)

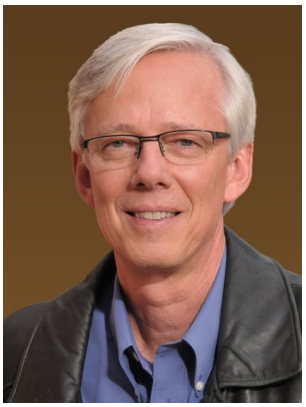
PMBOK: ©2013 Project Management Institute, Inc. All rights reserved

Requirements, user stories, RACI: International Institute of Business Analysis (IIBA), Business Analysis Body of Knowledge (BABOK) version 3.0 © 2015

Waterfall definition: https://quizlet.com/38088105/kpmg-flash-cards/

Wille, S., Kuehn, B., & Nelson, L. (2008). *Colorful leadership*. Portland OR: Eddlesen & Rowe.

Wille, S., Rairdon, J (2014). *Turn On All the Lights*. http://www.colorfulleadership.info/papers/human_ingenuity.htm



Steve Wille is an experienced information technology executive having worked for large corporations in many roles including software development, project manger, director of application development, vice president of corporate information systems, and senior vice president of a business division.

His book, *Colorful Leadership*, focuses on three perspectives of project management, people, process, and innovation. He has written and facilitated workshops on project management, business analysis, high performance project teams, and constructive conflict.

Steve is active in the Colorado IT community as a board member of SIM (Society for Information Management, president of RMIMA (Rocky Mountain Information Management Association), and an active member of Mile High PMI (Project Management Institute). His MBA is from Regis University and his BSBA is from the University of Denver.

