Introduction

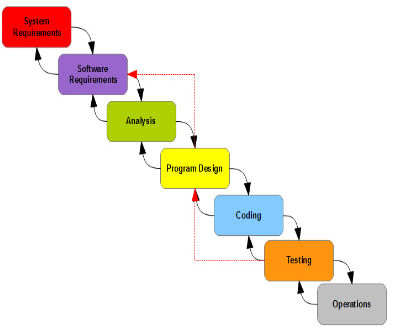
This article is a basic introduction to Agile software development methodologies and how to apply them. It is about how to work together to achieve a common goal.  This is not only suitable for software developers but also for Team Leaders, Project Managers, Product Managers, Development Managers, Testers, QA Managers, QA Engineers, Technical Writers, UX Designers, anyone involved in the delivering software.  This article focuses on how technology teams work together well to plan, build, and deliver software. It does not talk about code or specific technologies, or only about Microsoft tools. Hope this will improve your professional life and the effectiveness of your team.

The need for professional behavior: does our industry know what it means to behave? The definition of a software developer: who sits in a room, spends some time, and code comes out. We get very confused about deadlines, dates, estimates, and all of the things we are supposed to be doing, and we do them badly. Now that's not unusual. Our industry is still young.

Background

Winston Royce's Waterfall Model

"From the 1970 IEE paper "Managing the Development of Large Software Systems"



There are two essential steps common to all computer program developments, regardless of size or complexity. There is first an analysis step, followed by a coding step. Then we introduced the five most important steps:

Step 1: Program Design Comes First

Allocate processing, functions, design the database, define database processing, allocate execution time, define interfaces and processing modes with the operating system, describe input and output processing, and define preliminary operating procedures. Write an overview document that is understandable, informative, and current.

Step2: Document the Design

The first rule of managing software development is ruthless enforcement of documentation requirements.

Step 3: Do It Twice

The second most important criterion for success revolves around whether the product is totally original. If the computer program in question is being developed for the first time, arrange matters so that the version finally delivered to the customer for operational deployment is actually the second version in so far as critical design/operations areas are concerned.

Step 4: Plan, Control, and Monitor Testing

It is the phase of greatest risk in terms of dollars and schedule. It occurs at the last point in the schedule when backup alternatives are least available, if at all.

Step 5: Involve the Customer

It is important to involve the customer in a formal way so that he has committed himself at earlier points, before final delivery.

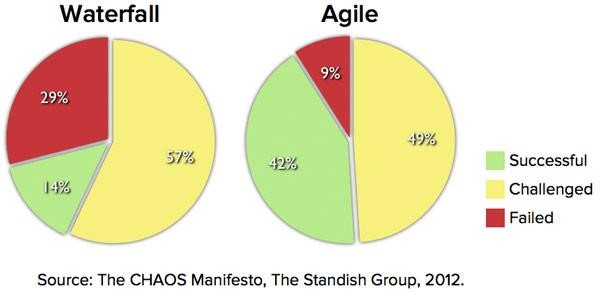
A careful reading of Royce's paper reveals:

Each phase should pass iteratively to the next

The entire process should be exercised twice before release

Royce knew that a single pass will fail

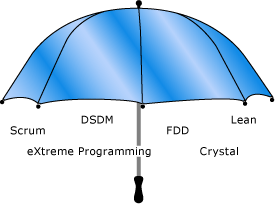
Unfortunately, for the process illustrated, the design iterations are never confined to the successive steps.



What are all this stuff?



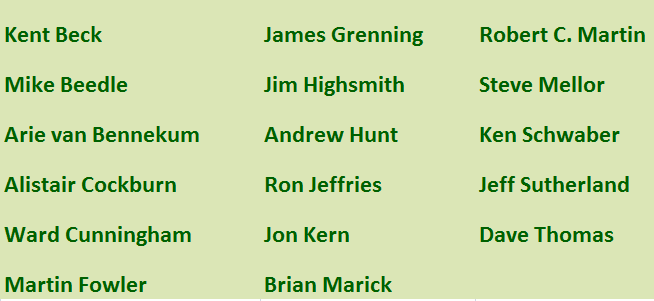
The answer is :



Agile development is not a methodology in itself. It is an umbrella term that describes several agile methodologies. At the signing of Agile Manifesto in 2001, these methodologies included Scrum, XP, Crystal, FDD, and DSDM. Since then, lean practices have also emerged as a valuable agile methodology and so are included under the agile development umbrella in the illustration later.

[To learn more, read Agile Principles and Values, by Jeff Sutherland](http://msdn.microsoft.com/en-us/library/dd997578.aspx)

Original signatories



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.

Twelve principles behind the Agile Manifesto

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

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

Business people and developers must work together daily throughout the project.

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

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

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

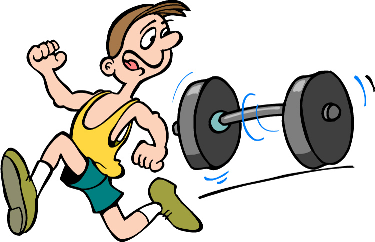
Simplicity--the art of maximizing the amount of work not done--is essential.

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

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

Please click [here](http://agilemanifesto.org/) for further detail on [Agile Manifesto](http://agilemanifesto.org/).

Many developers have lived through the nightmare of a project with no practices to guide it. The lack of effective practices leads to unpredictability, repeated error, and wasted effort. Customers are disappointed by slipping schedules, growing budgets, and poor quality. Developers are disheartened by working ever-longer hours to produce ever-poorer software.



DSDM

The DSDM (Dynamic Software Development Method) was developed to fill in some of the gaps in the RAD method by providing a framework which takes into account the entire development cycle. The main features of the DSDM method are as follows:

User involvement

Iterative and incremental development

Increased delivery frequency

Integrated tests at each phase

The acceptance of delivered products depends directly on fulfilling requirements

FDD

FDD is a wrapper methodology, in that it allows you to apply a method to manage projects at a very high level, but it still allows you to use other methodologies at a lower level. FDD's focus is on being able to set estimates and schedules and to report on the status of a project as a whole, or at a very granular level, but it doesn't prescribe a specific method to apply in order to create the schedule, leaving that up to you to decide. The idea is that you can look at your project and state with some certainty what the project status is, whether you are on time, slipping, early, and so on.

Lean

Lean Thinking is a way of approaching system optimization focusing on reducing waste and improving the overall flow of value through a system. Lean has a rich history in manufacturing and has gained popularity in software development circles in recent years.

Lean comes from Lean Manufacturing and is a set of principles for achieving quality, speed, and customer alignment. There are seven Principles of Lean Software Development:

Eliminate Waste

Build Quality In

Create Knowledge

Defer Commitment

Deliver Fast

Respect People

Optimize the Whole

Applying these principles to the work of delivering a software product is not an end goal. One is not said to "Do Lean"; rather one uses Lean principles to guide decision making and to choose techniques that will improve a system overall. For example, the practice of TDD (Test-Driven Development) builds integrity into software by inspecting it at the point of creation, thus supporting the Lean principle of building integrity during the creation process.

Plan

In Plan Driven Development a project is successful if it goes according to plan, so in software development it depends on the requirements stability, on having clear and fixed requirements. As you probably know, that is a luxury most software projects don’t have.

In plan-driven methodologies, it is less costly to change requirements during the design stage and it is more expensive to adapt to changes when construction has already started. So, a lot of energy is put into the planning phase. But software development is different.  There is no guarantee that a good design will make construction predictable.

"Walking on water and developing software from a specification are easy if both are frozen." - Edward V. Berard

The Agile approach is to break the dependency on requirements stability and come up with a process that takes into account changes. It does that by using Adaptive Planning and Evolutionary Design.

Adaptive planning implies going through the project cycle many times, re-planning, and re-adapting often.

Evolutionary design can be achieved with the help of practices like Self Testing Code, Continuous Integration, Refactoring, and Simple Design.

One is value-driven (Agile) and another is plan-driven (traditional). This is not to say that plan-driven approaches have no value; it is to say that in Agile, we make them explicit and discuss them often.

Both Agile and plan-driven approaches have situation-dependent shortcomings that, if not addressed, can lead to project failure. The challenge is to balance the two approaches to take advantage of their strengths in a given situation while compensating for their weaknesses.

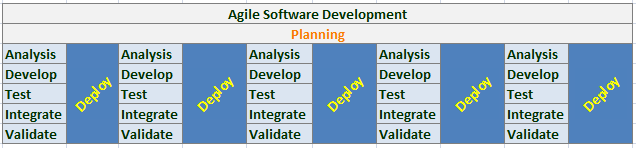
Plan vs. Agile

The fundamental difference between Plan driven development and Agile driven development lies between two significant differences. First one, in the Plan driven model the team will deploy one increment of software at the end of the project. But in Agile, the team will deploy a very small change of the software or more frequently. The second one is sequential verses concurrent activity.  In Plan driven development, a process starts after successful completion of another. But in Agile we plan all the time.

“Plan your work, then work your Plan” by Martin Fowlerand Neal Ford

https://secure-assets.readability.com/3a79ee0fa96b3ee1a2fa46771a1eb89bcf4243f1/687474703a2f2f7777772e636f646570726f6a6563742e636f6d2f4b422f6172636869746563747572652f3630343431372f506c616e56734167696c652e504e47

https://secure-assets.readability.com/759022199aaaa74c2d3c061f759642a670943ee0/687474703a2f2f7777772e636f646570726f6a6563742e636f6d2f4b422f6172636869746563747572652f3630343431372f506c616e56734167696c6554696d652e504e47



Both provide processes, tools, and techniques

Both require a disciplined approach to software development

Each has strengths and weaknesses

Agile methodologies address many situations for which Plan-driven methodologies are not well suited

Agile is still a disciplined approach to software development but with an emphasis on different aspects of the process!

Plan driven emphasizes formal communications and control

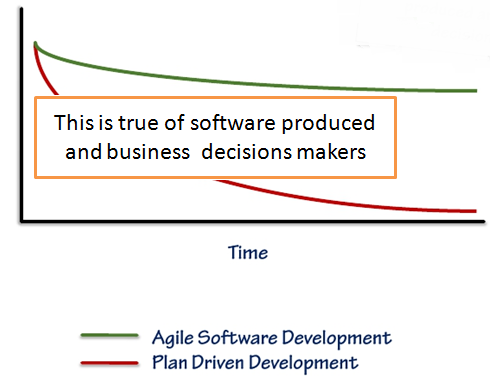
Agile emphasizes continual communication and ability to adapt to changes and uncertainty

Highly iterative to achieve quality over lots of gates to control quality

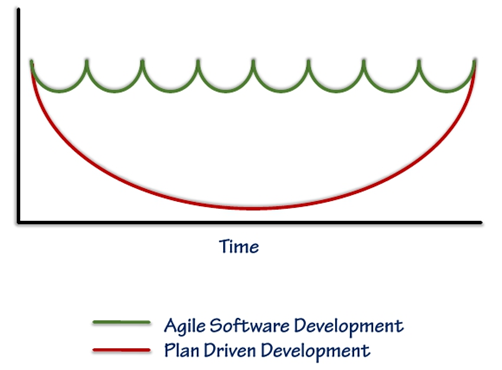
Inspect work as it is being done over Inspect product when it is complete

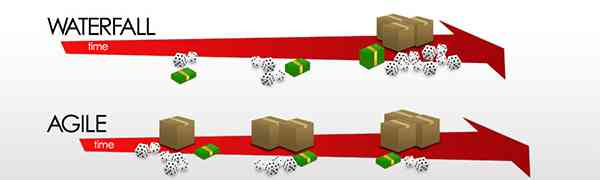
Start with a goal of filling a need over Start by predicting what will be delivered

Ability to changes



Visibility





Customer satisfaction by rapid, continuous delivery of useful software.

People and interactions are emphasized rather than process and tools.

Customers, developers and testers constantly interact with each other.

Working software is delivered frequently (weeks rather than months).

Face-to-face conversation is the best form of communication.

Close, daily cooperation between business people and developers.

Continuous attention to technical excellence and good design.

Regular adaptation to changing circumstances.

Even late changes in requirements are welcomed.

Management teams that work well with Plan-driven approaches also tend to work well with Agile approaches

However, management teams that lack the ability to work well with Plan-driven approaches may lack the focus required of Agile

Agile

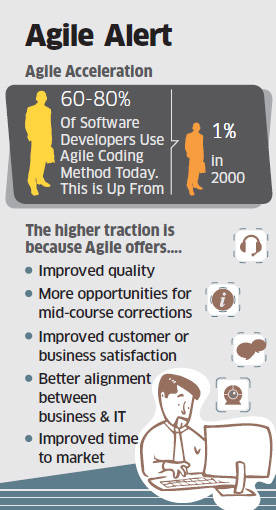
The principles and values of agile software development were formed as a way to help teams to break the cycle of process inflation and mainly focus on simple techniques for achieving their goals. The goal?? What is the the goal?

OK, the main goal of every software developer and every development team is to deliver the highest possible value to employers and customers. Yet our projects fail, or fail to deliver value.



The key is in Agile technique compressing the five sequences of the conventional software development method - called the Waterfall method - to a one-week cycle. It manages to do this by developing a system through repeated cycles (iterative) and in smaller portions (incremental), allowing developers to test and review during development. Speed, lower cost, and flexibility are key benefits.

"Adoption rates among IT departments are accelerating," says Phil Murphy, vice president and research director at Forrester in a 2011 report.



The participants in an agile process are not afraid of change. They view changes to the requirements as good things, because those changes mean that the team has learned more about what it will take to satisfy the customer. Agile team members work together on all aspects of the project. Each member is allowed input into the whole. No single team member is solely responsible for the architecture or the requirements or the tests. The team shares those responsibilities, and each team member has influence over them.

There are many agile processes: SCRUM, Crystal, Behavior-Driven Development (BDD), Test-Driven Development (TDD),Feature-Driven Development (FDD), Adaptive Software Development (ADP), Extreme Programming (XP), and more... However, the vast majority of successful agile teams have drawn from all these processes to tune their own particular flavor of agility. These adaptations appear to come together with the combination of SCRUM and XP, in which SCRUM practices are used to manage multiple teams that use XP.

Extreme Programming (XP)

As developers we need to remember that XP is not the only game in town.- Pete McBreen

Extreme Programming emphasizes teamwork. Managers, customers, and developers. It improves a software project in five essential ways: communication, simplicity, feedback, respect, and courage.

According to Wiki definition: "Extreme Programming (XP) is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements. As a type of agile software development, it advocates frequent "releases" in short development cycles, which is intended to improve productivity and introduce checkpoints where new customer requirements can be adopted."

Extreme Programming is a set of simple and concrete practices that combine into an agile development process. XP is a good general-purpose method for developing software. Many project teams will be able to adopt it as is. Many others will be able to adapt it by adding or modifying practices.

The ancestor of most Agile methodologies

Kent Beck created quite a buzz in the late 1990s and early 2000s

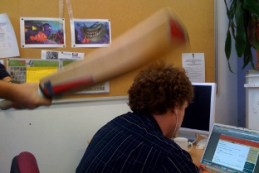
Blends processes and practices

Kent Beck’s basic idea

Take observed effective team practices

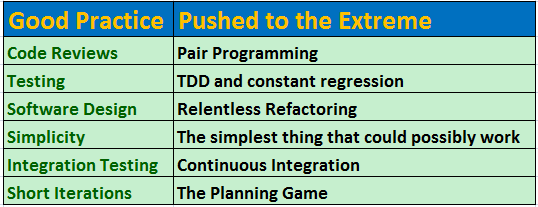
Push them to extreme level

What does it mean to say "Push them to extreme level"? Does that mean something like the following images?





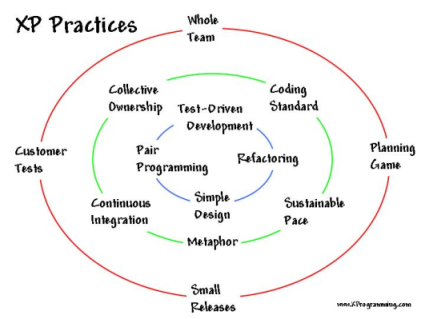
No, that doesn't mean XP. Let us see what that means.



XP is a set of practices that conforms to the values and principles of Agile. XP is a discrete method, whereas Agile is a classification. There are many Agile methods, XP is just one of them.

Having said that, none of the other Agile methods are as well defined, or as broad in scope as XP. Scrum, for example, is roughly equivalent to XP’s Planning game practice, with elements of Whole Team. While there are differences in the details, it is fair to say that Scrum is a subset of XP. Indeed, many Scrum teams augment their process by adding in many of the XP practices such as Acceptance Testing, Pair Programming, Continuous Integration, and especially Test Driven Development.

Of all the Agile methods, XP is the only method that provides deep and profound disciplines for the way developers do their daily work. Of those disciplines, Test Driven Development is the most revolutionary. Following are some good XP practices. I will try to write the details on each the next time.

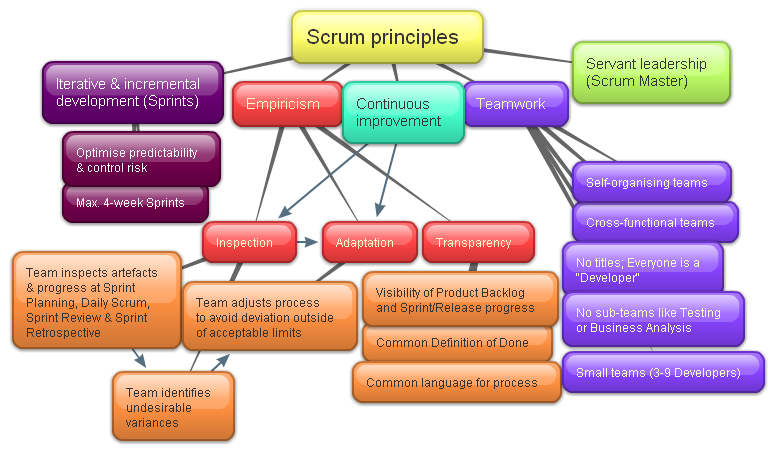


Scrum

Scrum is an iterative and incremental agile software development framework for managing software projects and product or application development. Its focus is on "a flexible, holistic product development strategy where a development team works as a unit to reach a common goal" as opposed to a "traditional, sequential approach“. Scrum asks why does it take so long and so much effort to do stuff. And why are we so bad at figuring out how long and how much effort things will take. Scrum embraces uncertainty and creativity. Because that is how people work. It places a structure around the learning process, enabling teams to assess both what they’ve created, and just as importantly, how they created it.

Scrum was first defined as "a flexible, holistic product development strategy in 1986 by Hirotaka Takeuchi and Ikujiro Nonaka.

In 1995, Sutherland and Schwaber jointly presented a paper describing the Scrum methodology. First public presentation.



Scrum Roles

There are these core roles for producing the product:

Product Owner

Development Team

ScrumMaster

Stakeholders

Managers

Product Owner

The Product Owner represents the stakeholders and is the voice of the customer.

Accountable for ensuring value to the business.

Writes (or the team) customer-centric items (user stories), prioritizes them, and adds them to the product backlog.

Scrum teams should have one, may also be a member of the development team.

Not be combined with that of the ScrumMaster.

Development Team



Responsible for delivering potentially shippable product increments at the end of each Sprint.

Made up of 3–9 people with cross-functional skills who do the actual work (analyze, design, develop, test, technical communication, document, etc.).

Self-organizing, even though they may interface with project management organizations (PMOs).

ScrumMaster

Accountable for removing impediments to the ability of the team to deliver the sprint goal/deliverables.

Is not the team leader, but acts as a buffer between the team and any distracting influences.

Ensures that the Scrum process is used as intended.

Enforcer of rules. Protector of the team and keep it focused on the tasks at hand.

Also been referred to as a servant-leader to reinforce these dual perspectives.

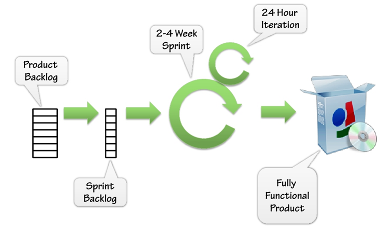
Differs from a Project Manager in that the latter may have people management responsibilities unrelated to the role of ScrumMaster.

Excludes any such additional people responsibilities.

Backlog

The product backlog is an ordered list of "requirements" that is maintained for a product. It consists of features, bug fixes, non-functional requirements, etc. - whatever needs to be done in order to successfully deliver a working software system. In Scrum, it is not required to start a project with a lengthy, upfront effort to document all requirements. This agile product backlog is almost always more than enough for a first sprint. The Scrum product backlog is then allowed to grow and change as more is learned about the product and its customers

The sprint backlog is the list of work the Development Team must address during the next sprint. The list is derived by selecting stories/features from the top of the product backlog until the Development Team feels it has enough work to fill the sprint. This is done by the Development Team asking "Can we also do this?" and adding stories/features to the sprint backlog.

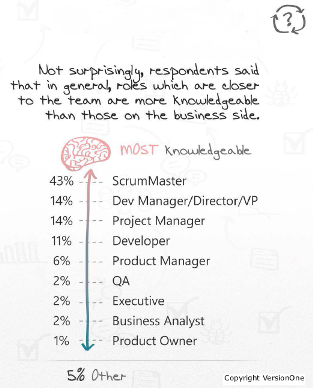


Conceptually, the team starts at the top of the prioritized Scrum backlog and draws a line after the lowest of the high-priority items they feel they can complete. In practice, it is not unusual to see a team select, for example, the top five items and then two items from lower on the list that are associated with the initial five.

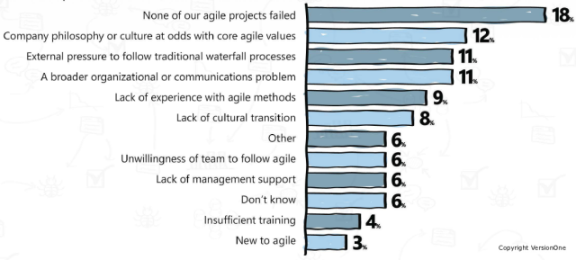
Agile Development Survey

The survey was conducted between August 9 and November 1, 2012. Sponsored by VersionOne, the survey polled 4,048 individuals from various channels in the software development communities. The data was analyzed and prepared into a summary report by Analysis.Net Research, an independent survey consultancy.

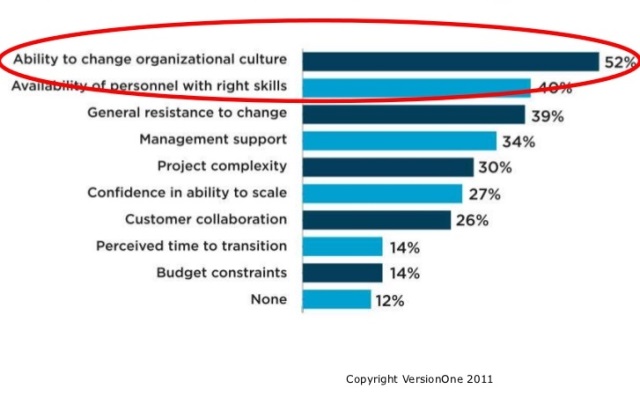
Who knows Agile?



Cause of Agile Failure



Barriers to Further Agile Adoption



Greatest Concerns about adopting Agile



Conclusion

A good agile team picks and chooses the management and technical practices that best work for them. When trying to adopt Agile practices, there will be a ton of excuses as why it won’t work. Those who understand the real benefits of the approach – and genuinely want to make the transition – will likely have success. Those who are searching for reasons why it will fail – well, they will likely find them and either abandon the effort entirely or end up practicing what Elisabeth Hendrickson calls fake agile.



In support of this conclusion, let me leave you with some words (Collected from Robert C. Martin):

"In preparing for battle I have always found that plans are useless, but planning is indispensable." - General Dwight David Eisenhower

It is better not to fixate on any given methodology, because the needs/conditions of the company and project are likely to change regularly, and you need to be flexible in how you approach managing projects if you want them to be successful.No single methodology is a silver bullet, so the trick is to determine which methods work for you and tune your methodology to suit your individual needs. This is what being "Agile" is fundamentally about.

Not Agile. It’s important to remember that Scrum and Agile methodologies are not a one-time look at a company's process, it’s an on-going philosophy based on continuous improvement.