UNIT-1
AGILE METHODOLOGY

Theories for Agile Management

What is Agile?

Agile is the ability to create and respond to change. It is a way of dealing with, and ultimately succeeding in, an uncertain and turbulent environment.

The authors of the Agile Manifesto chose "Agile" as the label for this whole idea because that word represented the adaptiveness and response to change which was so important to their approach.

It's really about thinking through how you can understand what's going on in the environment that you're in today, identify what uncertainty you're facing, and figure out how you can adapt to that as you go along.

Agile Software Development

Agile software development is more than frameworks such as Scrum, Extreme Programming or Feature-Driven Development (FDD).

Agile software development is more than practices such as pair programming, test-driven development, stand-ups, planning sessionsand sprints.

Agile software development is an umbrella term for a set of frameworks and practices based on the values and principles expressed in the Manifesto for Agile Software Development and the 12 Principles behind it. When you approach software development in a particular manner, it's generally good to live by these values and principles and use them to help figure out the right things to do given your particular context.

One thing that separates Agile from other approaches to software development is the focus on the people doing the work and how they work together. Solutions evolve through collaboration between self-organizing cross-functional teams utilizing the appropriate practices for their context.

There's a big focus in the Agile software development community on collaboration and the self-organizing team.

That doesn't mean that there aren't managers. It means that teams have the ability to figure out how they're going to approach things on their own.

It means that those teams are cross-functional. Those teams don't have to have specific roles involved so much as that when you get the team together, you make sure that you have all the right skill sets on the team.

There still is a place for managers. Managers make sure team members have, or obtain, the right skill sets. Managers provide the environment that allows the team to be successful. Managers mostly step back and let their team figure out how they are going to deliver products, but they step in when the teams try but are unable to resolve issues.

When most teams and organizations start doing Agile software development, they focus on the practices that help with collaboration and organizing the work, which is great. However, another key set of practices that are not as frequently followed but should be are specific technical practices that directly deal with developing software in a way that help your team deal with uncertainty. Those technical practices are essential and something you shouldn't overlook.

Traditional Model vs. Agile Model

Software development projects use different types of software development life cycle (SDLC)methodologies, depending on their nature and requirements, which basically define the way that the software development work is organized.

The two main approaches are the traditional, waterfall method and the agile software development method. How are they different from each other, and which one should you choose for your project?

The Agile Software Development Method Uses an Iterative and Team-Based Approach

One main difference between the traditional and agile methodologies is the sequence of the phases in which the software development project is completed.

The traditional method uses a linear approach, where the stages of the software development process must be completed in a sequential order. This means that a stage must be completed before the next one begins.

These stages usually comprise the following:

1. Requirements gathering and documentation
2. System design
3. Code and unit testing
4. System testing
5. User acceptance testing
6. Bug fixes
7. Product delivery

On the other hand, **the agile methodology uses an iterative and team-based approach**. Its main objective is to quickly deliver the application with complete and functional components. Instead of completing the software development tasks in sequence, they are completed in sprints that run from around one to four weeks and where a list of deliverables is completed in each sprint.

The tasks that do not get completed within the sprint are then reprioritized and included in future sprints. This also means that the different stages of the software development life cycle can be revisited as needed.

The typical agile approach involves the following stages:

1. Project initiation

2. Sprint planning
3. Demos

With the traditional method, the details of the entire project have been visualized and defined before the project starts. **In contrast, the agile methodology allows for more flexibility** in that changes can more easily be made even after the project starts.

It is best employed if the scope of the project cannot be clearly defined in advance. This also means that making unplanned software development changes with the traditional method is costlier than with agile.

The Agile Method Requires More Customer Involvement

Customers are highly involved in the early stages of the software development process when employing the traditional methodology. More specifically, their input is needed during the requirements gathering phase, as they must provide a detailed description of what their requirements are with regards to the software application to be developed and how they envision it to function.

However, they have limited involvement after the software development process starts, aside from attending status meetings, doing reviews, and providing approvals. They usually get to see the product in its entirety after a software development life cycle is completed.

In contrast, the customers are highly involved in every stage when employing the agile development process. They can review the application at every phase and make suggestions for improvement. As a result, the customers are more engaged in the entire software development process, in turn ensuring that they are satisfied with the finished product.

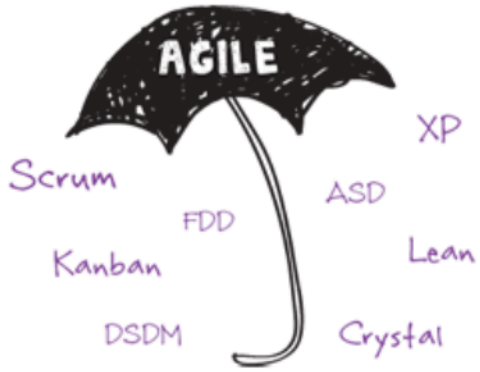
The Traditional Method Has a More Formal Documentation and Review Process

Each phase of the development process is properly documented and reviewed when using the traditional approach. On the other hand, due to the quick delivery time required with the agile method, changes are usually made directly on the code, with the developers just adding comments and annotations.

Classification of Agile Methods

What Is Agile Methodology?

The various agile Scrum methodologies share much of the same philosophy, as well as many of the same characteristics and practices. But from an implementation standpoint, each has its own recipe of practices, terminology, and tactics. Here we have summarized a few of the main agile software development methodology contenders:



Agile Scrum Methodology

Scrum is a lightweight agile project management framework with broad applicability for managing and controlling iterative and incremental projects of all types. Ken Schwaber, Mike Beedle, Jeff Sutherland and others have contributed significantly to the evolution of Scrum over the last decade. Scrum has garnered increasing popularity in the agile software development community due to its simplicity, proven productivity, and ability to act as a wrapper for various engineering practices promoted by other agile methodologies. With Scrum methodology, the "Product Owner" works closely with the team to identify and prioritize system functionality in form of a "Product Backlog". The Product Backlog consists of features, bug fixes, non-functional requirements, etc. - whatever needs to be done in order to successfully deliver a working software system. With priorities driven by the Product Owner, cross-functional teams estimate and sign-up to deliver "potentially shippable increments" of software during successive Sprints, typically lasting 30 days. Once a Sprint's Product Backlog is committed, no additional functionality can be added to the Sprint except by the team. Once a Sprint has been delivered, the Product Backlog is analyzed and reprioritized, if necessary, and the next set of functionality is selected for the next Sprint. Scrum methodology has been proven to scale to multiple teams across very large organizations with 800+ people. See how VersionOne supports Scrum Sprint Planning by making it easier to manage your Product Backlog.

Lean and Kanban Software Development

Lean Software Development is an iterative agile methodology originally developed by Mary and Tom Poppendieck. Lean Software Development owes much of its principles and practices to the Lean Enterprise movement, and the practices of companies like Toyota. Lean Software Development focuses the team on delivering Value to the customer, and on the efficiency of the "Value Stream," the mechanisms that deliver that Value. The main principles of Lean methodology include:

- Eliminating Waste
- Amplifying Learning
- Deciding as Late as Possible
- Delivering as Fast as Possible

- Empowering the Team
- Building Integrity In
- Seeing the Whole

Lean methodology eliminates waste through such practices as selecting only the truly valuable features for a system, prioritizing those selected, and delivering them in small batches. It emphasizes the speed and efficiency of development workflow, and relies on rapid and reliable feedback between programmers and customers. Lean uses the idea of work product being "pulled" via customer request. It focuses decision-making authority and ability on individuals and small teams, since research shows this to be faster and more efficient than hierarchical flow of control. Lean also concentrates on the efficiency of the use of team resources, trying to ensure that everyone is productive as much of the time as possible. It concentrates on concurrent work and the fewest possible intra-team workflow dependencies. Lean also strongly recommends that automated unit tests be written at the same time the code is written. The Kanban Method is used by organizations to manage the creation of products with an emphasis on continual delivery while not overburdening the development team. Like Scrum, Kanban is a process designed to help teams work together more effectively.

Kanban is based on 3 basic principles:

- **Visualize what you do today (workflow):** seeing all the items in context of each other can be very informative
- **Limit the amount of work in progress (WIP):** this helps balance the flow-based approach so teams don 't start and commit to too much work at once
- **Enhance flow:** when something is finished, the next highest thing from the backlog is pulled into play

Kanban promotes continuous collaboration and encourages active, ongoing learning and improving by defining the best possible team workflow. See how VersionOne supports Kanban software development.

Extreme Programming (XP)

XP, originally described by Kent Beck, has emerged as one of the most popular and controversial agile methodologies. XP is a disciplined approach to delivering high-quality software quickly and continuously. It promotes high customer involvement, rapid feedback loops, continuous testing, continuous planning, and close teamwork to deliver working software at very frequent intervals, typically every 1-3 weeks. The original XP recipe is based on four simple values €“ simplicity, communication, feedback, and courage €“ and twelve supporting practices:

- Planning Game
- Small Releases
- Customer Acceptance Tests
- Simple Design
- Pair Programming
- Test-Driven Development
- Refactoring
- Continuous Integration
- Collective Code Ownership

- Coding Standards
- Metaphor
- Sustainable Pace

Don Wells has depicted the XP process in a popular diagram. In XP, the "Customer" works very closely with the development team to define and prioritize granular units of functionality referred to as "User Stories". The development team estimates, plans, and delivers the highest priority user stories in the form of working, tested software on an iteration-by-iteration basis. In order to maximize productivity, the practices provide a supportive, lightweight framework to guide a team and ensure high-quality software.

Crystal

The Crystal methodology is one of the most lightweight, adaptable approaches to software development. Crystal is actually comprised of a family of agile methodologies such as Crystal Clear, Crystal Yellow, Crystal Orange and others, whose unique characteristics are driven by several factors such as team size, system criticality, and project priorities. This Crystal family addresses the realization that each project may require a slightly tailored set of policies, practices, and processes in order to meet the project 's unique characteristics. Several of the key tenets of Crystal include teamwork, communication, and simplicity, as well as reflection to frequently adjust and improve the process. Like other agile process methodologies, Crystal promotes early, frequent delivery of working software, high user involvement, adaptability, and the removal of bureaucracy or distractions. Alistair Cockburn, the originator of Crystal, has released a book, *Crystal Clear: A Human-Powered Methodology for Small Teams*.

Dynamic Systems Development Method (DSDM)

DSDM, dating back to 1994, grew out of the need to provide an industry standard project delivery framework for what was referred to as Rapid Application Development (RAD) at the time. While RAD was extremely popular in the early 1990 's, the RAD approach to software delivery evolved in a fairly unstructured manner. As a result, the DSDM Consortium was created and convened in 1994 with the goal of devising and promoting a common industry framework for rapid software delivery. Since 1994, the DSDM methodology has evolved and matured to provide a comprehensive foundation for planning, managing, executing, and scaling agile process and iterative software development projects. DSDM is based on nine key principles that primarily revolve around business needs/value, active user involvement, empowered teams, frequent delivery, integrated testing, and stakeholder collaboration. DSDM specifically calls out "fitness for business purpose" as the primary criteria for delivery and acceptance of a system, focusing on the useful 80% of the system that can be deployed in 20% of the time. Requirements are baselined at a high level early in the project. Rework is built into the process, and all development changes must be reversible. Requirements are planned and delivered in short, fixed-length time-boxes, also referred to as iterations, and requirements for DSDM projects are prioritized using MoSCoW Rules:

All critical work must be completed in a DSDM project. It is also important that not every requirement in a project or time-box is considered critical. Within each time-box, less critical items are included so

that if necessary, they can be removed to keep from impacting higher priority requirements on the schedule. The DSDM project framework is independent of, and can be implemented in conjunction with, other iterative methodologies such as Extreme Programming and the Rational Unified Process.

Feature-Driven Development (FDD)

The FDD variant of agile methodology was originally developed and articulated by Jeff De Luca, with contributions by M.A. Rajashima, Lim Bak Wee, Paul Szego, Jon Kern and Stephen Palmer. The first incarnations of FDD occurred as a result of collaboration between De Luca and OOD thought leader Peter Coad. FDD is a model-driven, short-iteration process. It begins with establishing an overall model shape. Then it continues with a series of two-week "design by feature, build by feature" iterations. The features are small, "useful in the eyes of the client" results. FDD designs the rest of the development process around feature delivery using the following eight practices:

- Domain Object Modeling
- Developing by Feature
- Component/Class Ownership
- Feature Teams
- Inspections
- Configuration Management
- Regular Builds
- Visibility of progress and results

FDD recommends specific programmer practices such as "Regular Builds" and "Component/Class Ownership". FDD's proponents claim that it scales more straightforwardly than other approaches, and is better suited to larger teams. Unlike other agile methods, FDD describes specific, very short phases of work, which are to be accomplished separately per feature. These include Domain Walkthrough, Design, Design Inspection, Code, Code Inspection, and Promote to Build.

The notion of "Domain Object Modeling" is increasingly interesting outside the FDD community, following the success of Eric Evans' book *Domain-Driven Design*.

Agile Manifesto and Principles

The **Agile Manifesto**, also called the Manifesto for *Agile Software Development*. The agile development manifesto represents the absolute cutting edge of the software development industry. It was created by representatives from Scrum, Extreme Programming, Adaptive Software Development, DSDM, Feature-Driven Development, Crystal and Pragmatic Programming.

The Agile Manifesto is created as an alternative to traditional heavyweight, document-driven software development processes such as the **Waterfall Model**. Agile software development focuses on keeping code simple, testing often and delivering functional bits of the application as soon as they're ready.

In Agile project management process works as a team plays an important role. It has been proven that learning in group is very effective & the human interaction helps to accomplish great things. Computers and Books are important but individual's interaction by sharing experience via debating, discussing and feeding off of each other's ideas is more important.

It has four key values and twelve Agile Principles to guide an iterative and people-centric approach to ***Agile Software Development***.

The values of the Agile Manifesto are:

1. Individuals and interactions over processes and tools

2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

Twelve Principles behind the Agile Manifesto helps to make clear what it is to be *Agile Software Development*:

1. Agile first priority is to fulfill the customer need from beginning to end and continuous improvement to ad into valuable software.
2. Agile allow change in requirements in the late in the development as well.
3. Agile works on delivering software regularly interval i.e. from couple of weeks to couple of month based on project.
4. Close daily cooperation between business people and developers throughout the project.
5. Key point is to trust, support and motivate individuals to get it projects build on time.
6. Daily face-to-face conversation is key point in agile testing. This is most efficient & effective way of communication.
7. Measuring progress by the amount of completed work.
8. Continually seeking excellence
9. Harnessing change for competitive advantage
10. Simplicity
11. Self-organizing team come out with best architectures, requirements, and designs.
12. Regular adaptation to changing circumstances with more effective way.

Agile Project Management

What is agile project management?

Definition

Agile project management is an iterative and incremental approach to delivering requirements throughout the project life cycle. At the core, agile projects should exhibit central values and behaviors of trust, flexibility, empowerment and collaboration.

- Read more in the blog: Agile project management - the what and the why.
- Common myths and misconceptions around agile
- Do you know your Scrum from your Sprint? Read our handy glossary of popular agile terminology to find out what they mean in the resources section below.

Agile Team Interactions

There are the three common roles found on an agile team:

1. Cross-functional team member–in keeping with the non-hierarchical structure of the agile team, the cross-functional team member is mentioned first, and NOT the product owner or team facilitator. They are professionals who deliver potentially releasable product on a regular basis. They need to deliver work in the shortest possible time, with higher quality, without external dependencies.
2. The product owner interacts with the customer and stakeholders as well as the team, and they pay attention to the highest value for the customer. They typically have a business background and have deep subject matter expertise. They create the backlog for and with the team, in a way that delivers the highest value without creating waste.

3. This role can be called various names, such as a project manager, scrum master, as well as a team lead, couch, or facilitator. The servant leader, no matter what he or she is called, needs to focus on facilitation of the work done by the team, impediment removal, and coaching. If the servant leader feels their internal coaching capability is not yet fully developed, then they may invite external agile coaches.

Ethics in Agile Teams

In addition to the Agile Manifesto, two of the most popular agile approaches, Extreme Programming (XP) and Scrum, advocate specific values. (Again, this is not to say that other software development approaches do not have values—just that XP and Scrum explicitly spell them out.) Ken Schwaber identified the following five Scrum values in his book *Agile Software Development with Scrum*: commitment, focus, openness, respect, and courage. Kent Beck identified these five XP values in his book *Extreme Programming Explained*: communication, simplicity, feedback, courage, and respect. While many trainers focus on the practices to get new agile teams up and running, it is the values that will keep the team truly agile long after the trainers and coaches leave.

Commitment: It's not enough to be assigned to a team and agree to try this new agile thing, and it's not enough simply to be dedicated. One must also be committed to doing whatever is necessary to meet the goals outlined and to take the authority to do so to heart. It means "to carry into action deliberately" (Merriam-Webster) or, as Yoda says, "Do, or do not. There is no try."

Focus: Don't get sidetracked. Remember what you committed to do, and focus your energies on fulfilling that promise. Distractions are not limited to the obvious things like email and unrelated meetings. They may also be things like creating documentation because "we've always done it that way" instead of creating documentation because it truly provides the customer with something of value.

Openness: Openness is about keeping the project status highly visible to everyone all the time. Anyone who is interested should be able to look at a wall, a wiki page, or a dashboard in an agile project management tool and see how many features have been completed, what's currently being worked on, and the goals of the iteration and release.Communication: People on the team must talk to each other. This gets harder to do the more geographically separated the team is, but with tools like Skype, teams can talk to one another easily and cheaply. The point is that team members must not rely on email and documentation alone. Verbal communication is required to clarify ideas, solve complex problems, answer questions quickly, and help team members coordinate work efforts.

Simplicity: Beck says the XP coach should ask the team, "What is the simplest thing that could possibly work?" Then they should do that thing. Because agile approaches develop code in increments each iteration, the thought is that it is better to develop something simple that may have to be expanded later if needed, rather than spend a large amount of time now developing a solution that's more complicated and may, in fact, not be necessary.

Feedback: As stated earlier, our number one obligation is to provide value to the customer. In order to do that, we must obtain frequent feedback from the customer or customer representative in order to make sure that the product we are building is meeting their expectations. And if it is not, we have the information we need now in order to make corrections. Beck extends this meaning of feedback to include feedback from the system itself in the form of unit, functional, and performance tests run frequently in each iteration.

Courage: In order to accept the authority and accountability for the delivery of the product, the team members all need courage. From the courage to make decisions to the courage to say no, this is a foundational value that gives rise to all the others.

Respect: Each team member must begin by agreeing that everyone deserves to be treated with respect. Many teams clarify this with working agreements that outline ways in which they choose to work together. And, as a result of working together to adhere to all the other values outlined here, team members grow to respect one another beyond their shared humanity to the ways each valued colleague contributes to the whole.

Adhering to a set of values that drive positive, collaborative behaviors and result in frequent software delivery helps us to work in an ethical manner. This is what agile is all about! Yes, agile also allows companies to decrease their time to market, increase quality, and eliminate waste in the process to provide a greater ROI to the organization. And you'll find plenty of articles about these benefits as well. But the realization of these benefits is not based just on the adoption of a set of practices; the value system must also be adopted if the approach is to work properly. So, as you consider adopting agile, also think about whether or not your organization's values match those of agile—and what you may want to do if they don't.

Agility in Design, Testing

Agile testing is a software testing process that follows the principles of **agile software development**. Agile testing aligns with iterative **Development Methodology** in which requirements develop gradually from customers and testing teams. The development is aligned with customer requirements.

Agile testing is a continuous process rather than being sequential. The testing begins at the start of the project and there is ongoing integration between testing and development. The common objective of agile development and testing is to achieve a high product quality.

Agile Documentations

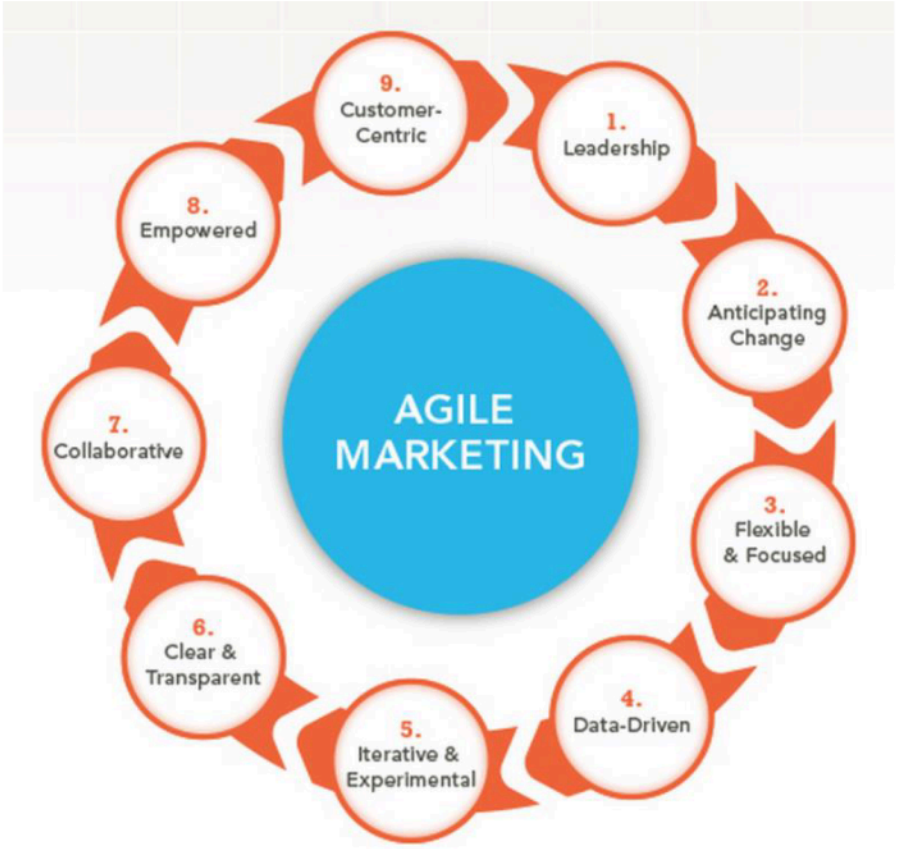
Ideally, an agile document is just barely good enough, or just barely sufficient, for the situation at hand. Documentation is an important part of agile software development projects, but unlike traditionalists who often see documentation as a risk reduction strategy, agilists typically see documentation as a strategy which increases overall project risk and therefore strive to be as efficient as possible when it comes to documentation. Agilists write documentation when that's the best way to achieve the relevant goals, but there often proves to be better ways to achieve those goals than writing static documentation. This article summarizes common "core practices" which agilists have adopted with respect to documentation.

Best practices for increasing the agility of documentation:

1. Writing
    - Prefer executable specifications over static documents
    - Document stable concepts, not speculative ideas
    - Generate system documentation
2. Simplification
    - Keep documentation just simple enough, but not too simple
    - Write the fewest documents with least overlap
    - Put the information in the most appropriate place
    - Display information publicly
3. Determining What to Document
    - Document with a purpose
    - Focus on the needs of the actual customers(s) of the document
    - The customer determines sufficiency
4. Determining When to Document
    - Iterate, iterate, iterate
    - Find better ways to communicate
    - Start with models you actually keep current
    - Update only when it hurts
5. General
    - Treat documentation like a requirement
    - Require people to justify documentation requests
    - Recognize that you need some documentation
    - Get someone with writing experience

Agile Drivers, Capabilities and Values

Nine Drivers of the Agile Advantage



1. Leadership

To put it bluntly, leadership can make or break The Agile Advantage. Leaders set the tone and create the type of empowered culture that transforms an organization. "It's all about culture. Your customer and process are important, but if you don't have a culture where your people feel empowered to be adaptable, to learn and communicate, you won't be able to serve the rest of the business," explains Leslie Snavely, VP of marketing and corporate business development at CHG Healthcare Services.

2. Anticipating Change

Anticipating the market is a benefit of being more agile with data and frequent research. By understanding customer patterns, you're able to foresee behaviors and get ahead othe competition. Elisa Steele, EVP of strategy and CMO at Jive Software puts it this way, "There are many times that no user told you that you needed to do [something], but you deeply know how the product works . . . you can actually do a leapfrog to innovate an idea, a program, an engagement opportunity that hasn't been done before."

3. Flexible and Focused

Flexible doesn't mean loose or sloppy, it means that like an athlete, you keep your eye focused on the ball but you're always ready to adapt – to sprint or turn on a dime if needed. "When we are focused we've been very successful in moving quickly and utilizing the right process to get things done," describes one lead marketer.

4. Data Driven

New tools for big data and analytics provide more intelligence on consumer behavior than ever before. You have to embrace data analytics in an actionable way, measuring early and often to discover customer patterns and responses. Leslie Snavely of CHG Healthcare describes their approach, "Data

shouldn't just be analyzed; it should be actionable. We use our data to make decisions because it's in a digestible form."

5. Iterative and Experimental

A4M means taking risks with new ideas and innovative methods. Support a culture that encourages taking chances, evaluating, iterating, and then re-evaluating. May Petry, VP of digital marketing at HP, says, "Agile enables me to test, learn, iterate and execute much quicker. If we didn't apply the agile lifestyle, I wouldn't be able to iterate and respond to change as quickly as I can right now."

6. Clear and Transparent

Operate with open visibility across teams as well as into goals, planning, performance and results. Get all teams working with clear, common objectives. Allen Olivo of Paypal advises, "If you want to move fast, you have to know where you are going. That's why our goal is to join people together around a shared purpose. This is the essence of what we want to accomplish."

7. Collaborative

A big change for many companies is that agility requires working across departments and roles in a flatter organization. Tom Vogl, CMO at The Clymb explains how they have done away with strict silos and hierarchy, "Being agile means having a strong degree of trust and respect across teams, In our case, people are not concerned about which group is going to get credit, but about what we're delivering and figuring out how to work together across teams."

8. Empowered

It's crucial to enable decision-making at all levels while allowing the freedom to fail. Tobias Lee, CMO at Thomson Reuters describes how this works for his team, "People feel empowered to share ideas and are supported to go do these things. And that's been very different, to feel like you can move fast with ideas you have because you're not concerned that management may not support you."

9. Customer-Centric

A4M companies put the customer at the center of all decisions, actions, and goals. "Knowing the customer, understanding the customer, talking to the customer, responding to the customer, and using their input to make marketing better. In short, agility is about responding to the market," shares Ann Lewnes, SVP and CMO at Adobe.

A company that adopts these Nine Drivers does not 'do marketing' in the traditional sense. After all, even traditional marketing has almost completely transformed. A4M apply these drivers to a mindset and methodology that empowers their organizations to better anticipate and respond to changes in the market, and in the process, outperform their competitors.