***Name: Atreya Mehta***

***Roll no: MA009***

***Subject:Pros And Cons Of Agile Methodology***

Agile product development practices have grown vastly in popularity and influence since their introduction more than twenty years ago. While these practices enjoy broad acceptance, there are certainly those who challenge and question the methodologies in practice while accepting the principles in theory. Some of these criticisms have merit while others are the result of poor training, planning, or implementation on the part of the critics and their organizations.  More often than not, the benefits of an Agile process outweigh the drawbacks, but to know whether Agile is “right” for your business, we first have to understand what it is and why it matters to us.

What is Agile and Why Does It Matter?



Agile development can be traced back to the lean manufacturing principles [W. Edwards Deming](https://en.wikipedia.org/wiki/W._Edwards_Deming) implemented in Japan as its manufacturing industry recovered from the ravages of World War 2.

Deming focused on reducing waste and empowering individuals to engage in constant improvement of the processes and procedures followed on manufacturing lines. The application of his principles to software development didn’t really begin to take shape until the 1990s, culminating in the 2001 creation of the [Agile Manifesto](http://www.agilemanifesto.org) during “Snowbird Conference.”

The Agile Manifesto and its corollary twelve principles describe in simple terms the underlying philosophy of Agile product development practices.The Agile Manifesto states:

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.

This statement of values was intended to represent the “new” concepts behind Agile approaches (“items on the left”) with the most-common waterfall practices permeated the software development world at the time (“items on the right”). The purpose of the Manifesto was to describe a new philosophy upon which more lean and adaptive software development practices could be built. Since its publication, many different methodologies have been built around these principles including [Scrum](https://en.wikipedia.org/wiki/Scrum_%28software_development%29), [Extreme Programming (XP)](https://en.wikipedia.org/wiki/Extreme_programming), [Kanban](https://en.wikipedia.org/wiki/Kanban_%28development%29), [Scrumban](https://en.wikipedia.org/wiki/Scrumban), and most recently [DevOps](https://en.wikipedia.org/wiki/DevOps).

It’s important, to understand Agile is a philosophy and not a prescribed or proscribed methodology for software development. In fact, it intentionally avoids proscription in its precise phrasing which accepts there is value in the “things on the right” (the traditional Waterfall model). At the same time, Agile principles value the “things on the left” more.

The goal of any Agile practice must be based on these principles, it must value individuals over processes, must focus on working software over documentation, must work in collaboration rather than against negotiated contracts, and most importantly must value responding to changing conditions and new information over blindly following a plan.

Many different proscriptive and prescriptive methodologies are built upon these principles. Scrum is probably the most common, with Scrumban or Kanban a close second. To implement these methodologies, successful teams must reflect upon and validate their efforts against the foundational principles of Agile development as a whole. Blindly following Scrum without adaptation does not value individuals over processes or promote responding to change over following a plan!

The Pros of Agile Development 

Responding to Change

Perhaps the biggest advantage Agile development practices bring to development teams and businesses in general is its emphasis on responding to change and focus on working on projects that matter when they matter. Agile methods don’t force us to attempt to divine the future in 9, 12, or 24-month projections. A properly-oriented Agile team has a list of the most important things they can work on; when they finish the most important thing on that list, they move to the next most important thing…and so on, ad infinitum. This type of focus has many benefits:

* Customers get solutions to the problems they value most, sooner
* Stakeholders can prioritize things in a progressive fashion reflecting actual market conditions at a given time
* Developers feel valued, since they’re working on things that actually matter and will receive frequent in-depth feedback from the very people using the product (ideally, at least).

Accepting Uncertainty

The second biggest source of value Agile development brings to an organization is that its practices accept the fact we don’t know everything about a project when we first start.This is in stark contrast to more “traditional” stage-gate and waterfall approaches, where requirements are expected to be “complete” before anyone so much as touches a keyboard to type their first line of code.

Agile instead accepts that we will discover more information as we go on; we may find a particular technical solution doesn’t meet the needs of customers, or we might discover there’s an entirely different problem underneath the stated problem, and by solving problem we can solve not just the proposed issue, but other customer concerns as well. Applying Agile principles to our approach allows us to accept the unknown and prioritize discovery and experimentation to drive out uncertainty before we fully commit to a solution.

Faster Review Cycles

In order for teams to be both accepting of uncertainty and responsive to change, there’s a need for rapid iteration and cyclical, comprehensive reviews as work is completed — to ensure that new discoveries are contemplated and current efforts are evaluated. Most Agile practices either time-box efforts (Scrum) or control the amount of “work in progress” (Kanban) to ensure efforts are completed within a reasonable amount of time. Those efforts are then reviewed with customers or customer proxies (such as internal services teams or stakeholder teams).

The focus on ensuring prompt reviews and feedback from actual users (or as close to the user as you can get) addresses the most common failings of waterfall approaches — the delivery of a product nobody really wants or likes after a 6-9 month closed development cycle.

Greater Flexibility in Releasing Features

In addition to faster cycles of review with customers or customer proxies, the focus on delivering working software in time-boxed or effort-boxed iterations of work provides the business as a whole more flexibility in when product should be delivered to end users.

In more traditional approaches, releases occur when all the planned work is completed…or, even worse, on a date set by stakeholders regardless of how polished the actual work is on that date. Agile approaches, on the other hand, provide sufficient functionality

Less Up-Front Work

Prior to the advent of Agile development methodologies, not only did product requirements attempt to predict what would be needed in 6-9 months, but they also attempted to be an encyclopedic contract outlining and detailing nearly every single aspect of a product’s design and development. It was not uncommon to see Product Requirements Documents (“PRDs”) and Technical Requirements Documents (“Specs”) exceeding fifty or more pages and outlining specific deliverables developers would use as checklists of exactly what to deliver — nothing more and nothing less.

Agile instead focuses us on defining and prioritizing problems to solve; collaborating with our developers to design, specify, and revise work needs to be done; and to exert only the amount of effort needed to move a product or project to its next phase. Incurring high up-front costs in investigation, documentation, and contract negotiation is anathematic to the fundamental beliefs outlined in the Manifesto.

The Cons of Agile Development 

Lack of Understanding

The biggest drawback of Agile development is most people don’t understand what it means to be Agile. As a result, they make unsupported assumptions about what it means for both development teams and the business. Many companies “want” to be Agile, but don’t invest the time, money, or effort to actually educate management or employees about how the principles apply and what methodologies will work best within their culture and organization. All too often, Agile is adopted as an inside-out solution…or worse, within a development team “bubble” that neglects the impacts on other business and product concerns.

Most of the criticism weighed against Agile in recent years has come from teams who have decided to “roll their own” solutions without taking the time to understand and at least attempt to implement the fundamental concepts and practices of the popular methodologies. Just as with any change in culture, process, or people, not investing in understanding why you’re making a change and the effects of said change will almost inevitably lead to failure.

Flexibility Can Lead to Bad Behaviors

Another common problem, leading from a lack of formal training or understanding is that the very flexibility of Agile as a philosophy can lead to teams engaging in bad behaviors, and “blaming” the resulting outcomes on Agile itself, rather than the dysfunctional choices made by the team. This tends to result from teams reading the Agile Manifesto and focusing solely on the “items on the left” to the exclusion of the “items on the right.”

No product can succeed without some level of process and tools, without some amount of documentation, without some negotiation of deliverables, or without some form of plan. When teams lean so far to the left of the list on the Manifesto, to the exclusion of the items on the right, they will eventually wind up engaging in behaviors result in bad outcomes. Further, the grassroots nature of Agile adoption in most companies actually allows these bad behaviors to not only establish themselves, but to spread as well — often when a company decides they’re “going Agile” they rely on internal people with “experience” to lead the change; if their experience is based on poor fundamentals, then their next Agile processes will suffer many of the same limitations.

Culture Fit

Not every corporate culture is “ready” for the changes Agile requires — which are not, as many pretend, relegated only to the development teams. The kind of flexibility, uncertainty, and regular interval-driven reviews that make Agile approaches successful require significant amounts of change throughout the entire organization:

* Stakeholders have to be available for review sessions, if not customers themselves.
* Marketing and Sales need to reduce their reliance on a predictable feature-based roadmap.
* Management and Executives need to understand Agile is a priority-driven “pull” model and not an opinion-driven “push” model.

All of these things require certain perspective shifts within the culture of an organization. If a company relies on 12-24 month roadmaps, regularly makes future promises to customers to secure current business, or requires budget approval before a project is started, they’ll be severely challenged in any attempt to maneuver into an Agile approach because their culture cannot support the assumptions upon which Agile success is based.

Lack of Predictability

Related to culture fit is the lack of predictability inherent in Agile approaches, which are a function of the acceptance of uncertainty and the focus on doing only what work is necessary to move to the next phase. When we accept uncertainty in our efforts and value responding to change over following a plan, we sacrifice the kind of predictability and security (often, a false sense of security) brought by feature-level 12- to 24-month roadmaps. This can often be a hard pill for the other parts of the organization to swallow; it complicates budgeting, marketing plans, sales comp plans, and even investor pitches.

If a company expects to be able to predict 6 months or more in the future with a high degree of certainty, they will absolutely be unable to achieve success using truly Agile means.

Integrating Diverse Skill Sets Into Teams

To be truly Agile, we often ask teams to be as fully capable of delivering quality results as possible within their team. Scrum, for example, theorizes every Scrum team [should be fully cross-functional](https://www.mountaingoatsoftware.com/agile/scrum/team) and able to deliver any end-to-end solution thrown at them. While this sounds very attractive in principle, it’s exceptionally difficult in practice, especially where things like design and testing are concerned.

Some software engineers have a knack for these things; some even have a knack for both things. All too often teams adopt Agile practices and they start to run into major issues figuring out how to work in the design and testing needed to deliver delightful products. Changing to a more Agile approach requires a strong level of discipline in leveraging automated and human testing as part of each work interval, and it’s something many companies struggle with. Design often becomes at worst an after-thought and at best a side-loading service delivery team, rather than an integrated part of each effort. These sacrifices can have real cost on the quality and capabilities teams can deliver.

Challenges at Scale

In recent years, there have been many attempts at building scalable Agile system architectures, such as LeSS. These concepts have been driven primarily by the difficulty of effectively scaling the work teams do into larger and larger organizations.

When the ideal size of a scrum team is most often positioned at between 5 and 9 people and your development team consists of 500 developers, how do you manage the interrelationships between those small teams while still maintaining a cohesive approach? Most Agile methodologies were designed for small, nimble,young organizations to adopt and adapt, but only in recent years have there been real efforts to identify and establish scalable Agile practices large organizations can apply.

Bottom Line: You Must Choose a Development Methodology, So Choose Wisely

Before making any significant changes to your development ideology or adopting one for your startup, you’ll want to look very closely at the principles and beliefs you and your organization hold about product development. You’ll also want to look at the realities of the product you’re actually going to be making.

If, for example, you’re working in a regulated environment and your product is going to require some kind of review by a Federal agency (such as the FCC), then you might not have the flexibility to adopt a truly Agile approach to your product design and development — what you send to the agency for review is what you have to build, by law. If, however, you’re building a SaaS platform and provides some B2C functionality, then you have lots of flexibility in when, why, and how you approach product development.

To be successful with Agile, you must accept uncertainty, value interactions, and focus on delivering working product every step of the way…and your stakeholders must understand both the costs and the benefits of those choices.