Global Knowledge®

Expert Reference Series of White Papers

12 Advantages of Agile Software Development

12 Advantages of Agile Software Development

Alan Koch, Global Knowledge Course Director, PMP, CSM

Introduction

There is a reason why the Agile methods are becoming mainstream. They can work! Although every Agile practice is not necessarily appropriate for every organization, each practice has delivered real value to many organizations, and some Agile practices can be used by anyone!

Come explore 12 ways in which the Agile methods are valuable. I'll bet that you will find more than a few that could be valuable for you!

Advantage #1: Customers' Needs Met

Agile projects involve the customer regularly, not just at the beginning (for requirements) and the end (for acceptance). This customer involvement mitigates one of the most consistent problems on software projects: What they will accept at the end of the project differs from what they told us at the beginning.

While good Business Analysis (or Requirements Analysis) practices can help with this risk, they can only take us so far. There is no substitute for demonstrating to the customer what we are building and getting their feedback regularly throughout the project. This is precisely what Agile projects do.

In addition to uncovering misunderstandings early in the project, this interaction helps the customer to form a better vision of the emerging product. Along with the ability to visualize the functionality that is coming based on having seen what was built so far, the customers develop a better understanding of their own needs and the vocabulary to express it to the developers. It also allows them to identify when their needs change (which we will discuss next in Advantage #2).

All of these dynamics come together to enable the customer to steer the project toward producing as much of what they need as can be done within the constraints of the project. (And we will address this topic of constraints in Advantage #3.)

Advantage #2: Greater Agility

The world does not remain static between the day we begin a project and when it is done. Whether the project takes a few days, several months, or more than a year, the organization changes, the customer changes, the environment changes, and yes, even the developers change.

The main reason why the Agile methods are called "Agile" is because the iterative lifecycle is designed to accommodate change. Work is done in short "iterations" (or Sprints) of only a few weeks, and the transition from one iteration to the next includes taking stock of what may have changed since the iteration began and how to adapt to those changes.

As we mentioned in Advantage #1, our customer's needs may change. It really doesn't matter whether that change constitutes the customer gaining a new and better understanding of their needs, or it is a result a very real changes in their environment. The bottom line is that delivering a product that meets the original (obsolete) needs is wasteful and counter-productive.

But the customer is not the only source of change. The development organization's situation could change as well. The changing business environment might impact the value proposition for the project, causing management to allocate more or fewer resources, rearrange priorities, extend or contract the timeline, or even suspend or cancel the project.

The iterative and incremental nature of the Agile planning process makes any of these sorts of changes much less disruptive than on traditional projects. Reworking the over-all project roadmap is relatively easy because it is based on rough order of magnitude estimates with very little accompanying detail. And because detailed planning is done just-in-time (for only a few weeks at a time), changes will cause little or no rework there as well.

Regardless of the source of the change, the customer is as involved in adapting to it as they are in any other part of the project. This guarantees that agility does not come at the expense of satisfying the customer (a point we will expand on next in Advantage #3).

Advantage #3: Realistic Customer Expectations

Most customers have little or no understanding of what it takes to develop software. This can result in many problems and arguments on projects as the customer makes demands that they imagine would be easy for the development team, and question where the time and effort is going and why the project is taking so long.

Agile projects include the customer in all of the most important activities. That is why the customer is counted as a member of the Agile team!

- The developers and the customer collaborate to define the high-level requirements (User Stories) and to maintain them throughout the project.
- The customer is present as the developers generate their rough estimates (e.g., Story Points) to answer questions about each requirement if necessary.
- The customer and the developers work out the order of development by considering the value of each feature to the customer as well as technical issues.
- The customer progressively elaborates the requirements details as the developers need them (mainly by responding to the developers' questions).

- The customer provides feedback to the developers about the product they are developing at least at the end of each iteration, and preferably more often.
- The customer and the developers collaborate to figure out how to adapt to each change as it is encountered on the project.

All of this interaction has the beneficial side effect of keeping the customer's expectations reasonable. Because of the high visibility into the developers' work, the customer quickly comes to appreciate the work the developers do and to respect them as the professionals that they are. When the customer's needs or understanding changes, they recognize that it will cost the developers time and effort, and they come to rely on the developers for estimates of those impacts.

All of this engenders a positive and collaborative working relationship between the developers and the customer. They quickly begin to operate as members of one team.

Advantage #4: Motivated Development Team

The positive relationship with a reasonable and satisfied customer is only one of the reasons why many developers prefer to work on Agile projects. The other main contributor is that they tend to value working in self-directed teams (which the Agile methods require for success).

Notice the bullets in Advantage #3, above. Agile projects are not planned by a Project Manager, the customer, an executive, or anyone else. The Agile team plans their own work, and then works their own plan. (And don't forget that the Customer is counted as a member of this team!) The dynamics are simple.

- The Customer makes sure that they end up with what is needed
- The developers figure out what it will take to make that happen
- Issues, complications, and trade-offs are discussed openly
- Compromises are negotiated and embraced by the team

In the end, the team (both developers and customer) own the plan and are responsible for its success. When corrective action needs to be taken, there are no fingers to point. "Our plan is wrong. We need to do something about it."

Self-directed teams have long been recognized and provide a good environment for any kind of knowledge-work. Software development is clearly knowledge-work, so it is surprising that it is so rarely done in self-directed teams. Most software developers appreciate the trust and respect that is implied when management empowers them to operate as a self-directed team.

Advantage #5: Productive Development Team

Of course, a motivated team (Advantage #4) is a productive team. But Agile teams are productive for reasons that go far beyond mere happiness! The Agile methods include several practices that enhance productivity.

On traditional projects, milestones tend to be few and far between. Agile projects have a significant milestone at the end of every iteration (every few weeks) — delivery of working software for customer acceptance. Because there is always a deadline staring them in the face, an Agile team can never afford to slack off. They are always driving toward a deadline that is relatively near.

Another productivity-enhancer is that Agile developers focus on working to "pay down" technical debt. Technical debt takes many forms on software projects, including these:

- Technical questions or unknowns that are left unresolved. Agile developers will attack these things early in the project to eliminate the uncertainty that can balloon into missed deadlines.
- Design errors that are left uncorrected. Agile developers will refactor working code to keep it malleable so they can continue incremental development, rather than coding around problems and postponing the day of reckoning.
- Testing left for later. Agile developers fully and completely test their code as they write it, and do not consider coding to be "done" until all tests pass. They do not count on testers or anyone else to catcher their defects, which can result in protracted testing phases at the end of the project.

In "paying down" technical debt, Agile developers make relatively small investments of time early in their projects to avoid the big timer-wasters later on.

Finally, the collaborative nature of an Agile development team means that each and every member of the team is learning from his or her peers continuously. Generalists become more generally capable, and specialists learn about each other's areas of specialty. In this way, each Agile project produces value that goes far beyond the product for the customer. Each and every team member becomes more and more capable with each Agile project.

Advantage #6: Refined Processes

Must of us recognize that a "Lessons Learned" exercise can be valuable. But how often do they happen on our projects? With rare exceptions, in spite of the value we know these exercises provide, somehow we can't make them happen on a regular basis.

Agile project teams hold a Retrospective (a mini-Lessons Learned) at the end of each iteration of every project. Because it is just a natural part of the process, it does not get skipped or preempted. And because it happens regularly, Agile project teams derive greater value from their retrospectives than organizations that actually do a Lessons Learned on every project.

As a part of the transition from one iteration of work to the next, the Agile team will ask themselves these questions:

- What worked well for us in this iteration? How can we be sure to keep doing it?
- What was problematic for us in this iteration? How can we fix it?
- What do we not understand about something in this iteration? How can we learn about it?
- What did we do differently in this iteration than before? Should we keep the change or go back to the old way?

In other words, Agile teams are engaging in the most effective form of continual process improvement that you are likely to see in the software world.

The net result of this simple exercise every few weeks is that an Agile team will very quickly refine their processes. Ineffective practices will be replaced with better ones, effective practices will be strengthened, and problems will be solved.

Advantage #7: Good Quality Software

Regardless of how you define quality, Agile teams will deliver it. Some examples of how quality is defined include these:

Fitness for Purpose. The regular and continuous interaction between the customer and the developers (discussed in Advantages #1 and #3 above) have as their primary objective assuring that the product as built does what the customer needs for it to do. As long as the customer is effectively participating in the team, the developers will produce the right product!

Fitness for Use. Again, the regular and continuous interaction between the customer and the developers (discussed in Advantages #1 & #3 above) assures the usability of the product as well. In addition, Agile projects assure usability by releasing the product into actual use as early and as often as possible during the project. Nothing confirms usability better than the actual users trying it out!

Lack of Defects. The strong technical focus (discussed under Advantage #5, above) results in much better testing on an Agile project than in most other methods. As mentioned above, Agile developers take responsibility for the quality of the code they write. In addition to producing cleaner code, it means that if there are testing specialists on the project, they will start their testing with better software, which always results in more effective testing and a better resulting product!

Reliability, Maintainability, etc. Again, the strong technical focus (discussed under Advantage #5, above) results in a technically superior product. In addition to the focus on testing and refactoring, some Agile teams use practices like coding standards, peer reviews, and pair programming to assure that the code they produce is technically solid.

Conformance to Specification. Agile teams subscribe to many Lean principles including these two:

- Decide as Late as Possible. Postpone commitment to a decision until the latest responsible moment.
- Deliver as Fast as Possible. When a decision has been made, act upon it right away.

These Lean principles keep specifications to a minimum. The Agile team makes commitments only when those decision need to be made, which is generally at the point of implementation. In this way, Agile teams assure that conformance will not be a problem.

Advantage #8: Improving Estimates

Many software developers are notoriously poor at estimating their work. It is not uncommon for Project Managers who are creating plans to ask their developers for estimates of the work, then to double or triple those estimates in the plans.

Because of the short feedback loops in an Agile project, developers can begin to learn about their estimating errors and become better at this crucial task. The two main forms that this learning takes place are these:

At the beginning of the project, each requirement (User Story) is given a rough order of magnitude estimate (e.g., Story Points) in a team workshop. As a part of this exercise, the team members discuss what is being estimated and agree together on the estimates. Then, as part of the transition from each iteration of work to the next, one of the changes the team might embrace is the need to re-estimate. They would do this if it became clear to them that the original estimates (which they all participated in) were bad. The re-estimation exercise provides them with an opportunity to learn what they had done wrong the first time.

The second opportunity to learn comes from the more detailed estimating of the actual work that is done at the beginning of each iteration. Here, it is the short feedback loop that comes into play. They estimate the work they will do over the next few weeks, and they learn how accurate those estimates were in short order. Then in a few weeks, they are estimating the next iteration. Each Agile project provides the team members with opportunities to hone their estimation abilities many times over.

Advantage #9: On Time & On Budget

The Agile methods all make the assumption that the project timeline (the project Time-box) is inviolable. In addition, they assume that the main driver of budget is people's time, so the project time-box implies a fixed budget as well. Good Project Management practice tells us that you must have a variable to manage in order to have a successful project. In the Agile methods, that variable is product Scope and Requirements.

An Agile team will always deliver on time and on budget. The only question concerns precisely what will be delivered! This is the reason for the close participation of the customer in the project. That person's role is to ensure that the project meets their needs (Advantage #1) to the greatest possible extent within the project constraints (as mentioned in Advantage #3).

Often the reality is that the customer's true needs (minus gold-plating) can be met within a constrained timeframe or budget. Of course this doesn't guard against a really ill-conceived project that cannot produce meaningful results as constrained. But that is addressed in Advantage #10.

Advantage #10: Early Warning of Problems

The traditional project planning process is supposed to help us identify when a project cannot be successful before the work begins. But in reality, the number of unknowns often makes it impossible to make such dire predictions with any certainty. This is just as true with Agile projects as with any other. But Agile projects have

an advantage in that there is a ready-made barometer built into the process that will provide early warning of impending doom.

Again, it is the iterative development pattern that provides this warning. After the team has laid out a project roadmap based on their best assumptions about technical issues and the rate at which they can work, they immediately begin producing working software.

After a few weeks, the first iteration is complete, and they deliver software. If it is significantly less that anticipated, an early flicker of red begins to show. After a second iteration delivers less than promised, the red light is clearly visible. If they continue for a third iteration and again deliver under plan, the red light is flashing and the siren is blaring.

Less than halfway through a small project (less than 10% of the way through a big one), everyone knows without doubt that the original project constraints (timeline or budget) or assumptions were out of line with what is possible. This early warning provides plenty of time to renegotiate expectation(s), re-adjust plans, or cancel a doomed project before any more time or money is wasted.

Agility doesn't guarantee success. But failure is clear much sooner that with other approaches.

Advantage #11: Meaningful Milestones

As we discussed in Advantage #5, Agile projects have milestones much more often than most traditional projects. In addition to being more numerous, Agile projects' milestones are also more meaningful.

Consider the types of milestones traditional projects rely upon: Requirements sign-off. Critical Design Review. Code complete. Testing complete. Customer Acceptance. Most of these milestones only have meaning to people who understand the software development process. And those of us who fall into that category (if we are honest with ourselves) recognize that (with the exception of the final one) these "milestones" are artificial measures of progress. They are nothing more than indications that we have done certain work that should have taken us a step closer to successful project completion.

Contrast that with the milestones on Agile projects: Customer Acceptance. Customer Acceptance. Customer Acceptance. Customer Acceptance. … At the end of each iteration of work (every few weeks), the project produces actual working software that the customer can evaluate and either accept or not. (And if the customer has been engaged with the developers as they did the work, why would it not be acceptable?) And if these milestones are missed – we addressed that in Advantage #10, above.

What more meaningful milestone can you have than to deliver what the customer asked for and having them accept it?

Advantage #12: Management Visibility

Managing software projects can be frustrating. There is a tremendous amount of activity going on that we hope is driving toward success, but surprises and unexpected problems and changes often spring from nowhere to foil the best of plans. The reason for these unhappy surprises lies in the artificiality of our measures of progress, as discussed above in Advantage #11.

Agile projects provide tremendous visibility and transparency to any manager who understands what he or she is seeing. Understating is an important part of this because Agile projects look quite different from traditional projects! With appropriate tutoring, a Manager will be able to see:

- The project's roadmap and what it means
- Changes to that roadmap and their implications
- Early indications of customer satisfaction (see Advantage #1)
- Early warning of problems (see Advantage #11)
- Impact of changing the project's budget or schedule

With this level of visibility, managers will actually be able to manage software projects, as opposed to hoping for the best.

What? Your Agile Projects Don't Look Like This?

If what you see on the "Agile" projects in your organization doesn't look like what is described in this paper, you are not alone. Too many teams claim to be using an "Agile" approach, when they are merely throwing off the chains of discipline.

As was so well said by Kent Beck (the author of *Extreme Programming Explained* and creator of Extreme Programming – XP), "Agility requires iron discipline!" Get your team the help they need to start actually operating in an Agile way, and you will not only start realizing the 12 Advantages discussed here, but your developers will thank you as well!

Learn More

Learn more about how you can improve productivity, enhance efficiency, and sharpen your competitive edge. Check out the following Global Knowledge courses:

Agile Boot Camp

Agile Project Management

Collaborating and Communicating Agile Requirements

For more information or to register, visit **www.globalknowledge.com** or call **1-800-COURSES** to speak with a sales representative.

Our courses and enhanced, hands-on labs and exercises offer practical skills and tips that you can immediately put to use. Our expert instructors draw upon their experiences to help you understand key concepts and how to apply them to your specific work situation. Choose from our more than 1,200 courses, delivered through Classrooms, e-Learning, and On-site sessions, to meet your IT and business training needs.

About the Author

Alan S Koch, PMP, CSM, is a Global Knowledge Course Director, author, speaker, and consultant Alan and his company, ASK Process, Inc. (www.aksprocess.com), have enabled thousands of engineers and managers to realize IT project success. Read his book, *Agile Software Development: Evaluating the Methods for Your Organization*, to learn how you can use the Agile methods to achieve project success!