

Corporate Technology Solutions, Inc.

Agile Software Development

An Executive Summary by Cobbie Behrend

Table of Contents:

WHY AGILE?	3
VALUE:	3
CONTINUOUS USEFULL FEEDBACK:	4
IMPLEMENTING AGILE:	4
CONCLUSION:	5

Why Agile?

Agile is a commitment to value above action for action's sake. Yet many software development practices, even some that are labeled as "agile", are adopted for reasons other than value, such as:

- * corporate tradition
- * ceremony
- * personal preference
- * corporate mandate

Reasons such as these have no inherent defect. However, when they are the sole motivation for actions that carry at most an anticipation of value they are not enough to guaranty long-term success. Organizations must conduct frequent assessments that promote solutions to roadblocks and issues in order to ensure that activities are providing value. On the organizational level you can think of agile as:

We work to provide value on the basis of continuous useful feedback.

Although it is the promotion of value and continuous useful feedback that gives agile its power, it is important to also remember that this may not lead to the same practices in all organizations.

Value:

If you are part of a software development team, you provide value to your organization by creating working software that meets customer needs. Things that interfere with providing working code reduce your value. For example, documentation and design are important parts of developing software, but their value is lost if you are documenting the wrong solution, or a solution that is riddled with bugs. On the other hand, if you have no software documentation and developers are unable to maintain your software in a timely manner, then the value of your software is also reduced. In the other extreme your documentation has little value if it is so copious that it becomes difficult to maintain.

Beyond the value of end products, teams and organizations must also measure the contributions of their software development practices. Your project can save your organization time and money, and increase the popularity of your software, but how

much your development methods improve from release to release matters, too. If you measure the effectiveness of development methods your future successes become more predictable and impending failures easier to spot, which maximizes your efficiency. If you measure only once, analyze, and adapt, you may encounter some success. To achieve continual success through greater predictability and efficiency you must measure continually.

Continuous Useful Feedback:

The grease that allows the wheels of continuous feedback to turn freely is open collaboration and interaction between all participants. Team interaction is facilitated by practices such as automated builds, code reviews, stand up meetings, and pair programming. For the organization frequent releases, release reviews and critiques, user involvement, stake-holder decision making and prioritization, and removing development impediments are helpful collaboration practices.

All agile methodologies revolve around the notion of iterative development. In agile, software is built in iteration or releases. Each release is also fully functioning relative to the requirement it sought to implement. Releases are generally 2-6 weeks depending on the features being implemented. An iteration will result in lessons learned and metrics that will be used in the upcoming iterations. A project manager will use the data to better estimate the remaining efforts. While an architect will use the same data to better understand the software complexity and better understand the risk areas.

Whatever feedback you produce, it is important that you ensure such feedback is useful. If the feedback the organization receives is not useful, you need to examine the underlying practices and change them. Even when you prove feedback to be useful, you must strive to continually improve processes to ensure that it stays useful. For example, the length of the feedback loop, how long it takes from when an action occurs to when you find out the value of your action, is critical to the usefulness of that interaction. You may need to shorten or lengthen feedback loops to maximize benefits as individual, team, or organizational needs change.

Implementing Agile:

Agile is a simple concept, but implementing it is often challenging. It is difficult to find what works for a given team, individual, organization, and even more difficult to find out why something didn't work. Often you hear about a team in an organization that had great success using agile, but were not able to replicate that success to other teams in

their organization. To understand why this is so consider living your own life in an agile manner:

I work to provide value on the basis of continuous useful feedback.

If you have too much feedback to process you will never get anything done. However, too little feedback and you reduce your effectiveness. You can see how the balance between just the right amount of useful feedback and providing the most value possible is not obvious. The fact that teams have more than one person who are all different, and that organizations have multiple teams who have different strengths, serves to make this problem even more difficult. Organizations can meet this difficulty by deploying capable coaches to their agile teams who can guide teams toward practices that enhance and build on existing strengths.

Conclusion:

Agile software methodologies emphasize continuous useful feedback instead of “in the dark” development. This is facilitated by many agile advocated practices such as stand-up meetings and code reviews. Most notable of these practices is the short development life cycle that produces working software. At the end of each development cycle the team is armed with information that they can take into account when planning the next iteration.