

# Agile? Waterfall? How About Wet Agile?

## The Journey to Full Agile Development Often Begins with a Hybrid Approach

Have you noticed how some organizations claim to be Agile ... but when you peel back the layers they're really a Waterfall shop with aspirations to become Agile?

For the last three years, I have been working for a Chicago-based IT consulting firm that specializes in custom software development. As a consultant, I've found that we often end up using the methodology that is in place at the client site but, when given the choice, I usually chose Agile over Waterfall methods. However, the reality is that most development organizations are still Waterfall shops.

So why do so many organizations aspire to become Agile yet are still only implementing a few aspects of it? Most likely, they're not given enough time to plan the process of becoming Agile. Or, it may be because Agile is usually introduced by the development community in a bottom-up style that isn't understood or appreciated from the top-down. Regardless, many teams end up implementing a hybrid methodology.

Hybrid approaches to solving problems are not all bad. Case in point: The auto industry needs to find alternatives to high gas prices and knows that the market is ready for an electric car. However, they started with a hybrid car, known as the Toyota Prius, before introducing a full electric version like the Chevy Volt. So perhaps the reality on how we make that quantum leap to full Agile development starts with a journey that begins with a hybrid approach.

I created a new term to describe Waterfall shops that either aspire to be Agile or are in the transition process. I call these shops "WetAgile." Individual people can also be described as WetAgile. For example, perhaps you're an aspiring Agile guru doing so many Waterfall projects that you find yourself implementing the pieces of Agile that you most understand or believe your Waterfall team can easily embrace.

When I look back at some of my own most productive teams, they all had one thing in common: Each project team was using one or more aspects of Agile.

## Paired Programming Works Great for Both Newbies and New Technologies

For example, we recently completed a large electronic content management (ECM) implementation for a client. The team consisted of highly skilled Java developers who had just completed product training. Soon after their training class, the developers moved on to their programming tasks but had difficulties figuring out how to wire it all together. They quickly discovered that working together was faster than working independently. Although I would like take credit for telling them to work together, it was something that they tried on their own. They started with one developer searching the product knowledge base while the other developer implemented code fragment. Within a few days, they were sitting side-by-side, coding and refactoring each other's code. In the end, paired programming techniques allowed them to complete all of their deliverables faster than if they had worked independently of each other. I quickly became a proponent of paired programming and have since encouraged subsequent project teams to give it a try.

The traditional approach to bringing a project newbie up to speed is to ask them to read technical documentation and ask them to check back in a few hours later. Hands-on learning via paired programming and having someone available to answer questions is far more productive in the long run because most of us learn faster by doing than reading.

## Bite Size Chunks Go a Long Way

Short development sprints can also be very productive. Too often we start a project by trying to estimate when it will all be done. Since I've seen so many projects slip two to three months beyond their original estimated completion date, I have become a fan of short sprints, or, bite size chunks.

Short sprints include regular development build and releases that continue until the business sponsor agrees that the cycle is complete. Depending on the size of the project, weekly development cycles (or in some cases bi-monthly releases) is preferable. These short sprint cycles are easy to implement and readily embraced because progress is transparent and large projects can be broken down into bite size chunks. And most importantly, if you're fully engaging the business in the development cycle, the immediate feedback is invaluable.

Continuous feedback between the business and the project team provides visibility into the progress of the project and is also instrumental in course correcting the team. We've all seen what happens when feedback doesn't exist: "That's what I asked for but not what I wanted."

## Getting it Done, Done, Done

Once down the Agile path, it doesn't take long to start appreciating the Agile term "done, done, done." The first time I heard *done, done, done*, I thought my friend invented it. However, I later learned that it's an Agile term that really resonates with development teams.

*Development Done, QA Done and Business Done* is easy to explain and something that everyone understands. Too often the developer calls a task done and progress is reported to an oversight team. The oversight team interprets this to mean that the end result is near when, in fact, there are two more levels of done to achieve. We all want to deliver the good news that we're making progress on our tasks and the end is in sight. As a result, we often prematurely call something done when it is only partially done. So the next time a developer tells you their tasks are done, ask if QA or the business would agree. Once everyone understands *done, done, done*, and we're all using the same terminology, communication is improved. This has a positive impact on the team culture and business results.

Because the word "done" can have different meanings for different people, it's important that IT and the business establish a shared common language to improve communication and foster better relationships.

## The Business Value of Storyboards

The Agile process of storyboard sessions with the business users is extremely valuable and something that really resonates with the business. Storyboarding is a process by which the business and IT collaborate on system and screen designs. Utilizing paper post its, system interfaces are laid out in a manner that helps capture the intent of what the business wants. Regardless of complexity, every attempt should be made to storyboard all user interface screens in order to remove as much ambiguity as possible *before* the development cycle begins.

For example, a recent client needed to design a marketing program that included a new user interface. The project team consisted of approximately eight business users and a handful of representatives from IT. We broke the team into two groups of four and paired them up with a business analyst. The teams were then asked to use blank sheets of paper and draw screens which included pick lists, drop down values, text boxes and links to other screens. Next, they placed the screens in the order of how they envisioned them flowing. When the teams completed the exercise, we taped the screens onto a white board in the specific order with one design above the other. The two teams then reviewed both designs along with a group discussion.

In just two hours, we reached agreement on a set of screens that we could hand over to the IT team to begin breaking down requirements and estimates. This seemingly simple process of storyboarding the user interface gave the IT team clarity that was needed to accurately estimate the size and effort of what they wanted. All-in-all, the storyboarding process was highly effective in reaching alignment with the business because it provided valuable insight for the overall vision.

While engaging the business in face-to-face meetings to flush out design is productive, it can also get tricky. Typically, everyone embraces the idea of just-in-time design or just-in-time requirements. However, as the project progresses, the business frequently becomes less available to participate in design sessions and the methodology begins to breakdown. The trick is to keep the business engaged during the storyboarding exercise and focus on capturing the intent of what they want built.

Don't attempt to storyboard everything in one session. Rather, break the work into manageable pieces and you'll position the team for more successful outcomes. Half-day sessions are much more effective than full day sessions because the group will either lose focus or begin to worry that they are away from their regular jobs for too long. If you can manage to keep the business engaged during the white boarding exercise you'll find it easier to keep them engaged through the remainder of the project.

### **Business Integration Builds Trust**

The most important Agile concept to introduce to the non-Agile team is tight integration with the business. This is achieved by leveraging the above techniques and close examination of your organization's requirements definition and management practices.

One way to measure whether you've achieved the required level of integration is to evaluate the trust that exists between IT and the business. If the necessary level of trust is not there, take the necessary steps to improve the relationship. Characteristics of a highly trusted relationship include increased levels of business productivity and a feeling as though the project team has captured the intent of the business need. Once intent is captured, the relationship will begin growing in a positive direction.

Regardless of the techniques you use, the end goal should be to predictably deliver business results and avoid an IT driven solution. Too often, the business asks IT to solve a problem by a particular date. Then, they let IT design the solution only to comment later, "That's not what I wanted!"

### **Conclusion**

One word of caution - if you're going to implement pieces of Agile, make sure the executives and stakeholders understand that you're not claiming to be an Agile shop. Rather, you are picking and choosing the pieces of Agile that fit best in the organization. If you're going to swap out a component of Waterfall with an Agile practice, make sure that it supports the overall goals of the project and you fully understand the role of what you're swapping out.

For example, if you decide to not implement paired programming you may need to increase the amount of testing hours. Or, if you decide to not implement code refactoring during a development cycle because the overall design is not extendable, you may need to add additional time to the end of the project or an additional phase of the project after the production release.

All in all, the play list of the best of Agile includes short development sprints; paired programming; done, done, done; story boarding and integration with the business. Before beginning your next project, consider how Agile techniques can help you achieve the ability to:

- Show and measure progress through short iterations
- Use multiple heads to solve problems
- Maintain a healthy relationship between IT and the business
- Make sure the business agrees that the project is done

Think of WetAgile as the ala-carte method of picking the components of Agile that you either understand or think you can easily implement in a Waterfall shop. Keep in mind that as you take on more projects, you'll most likely be able introduce more and more concepts of Agile and eventually become the Agile shop that you aspire to become.

Somebody once asked me "have you managed an Agile project team?" to which I responded, "I did one better, I managed a WetAgile team."

### About the Author

Steve Pieczko has more than thirteen years experience managing custom software development and implementing packaged products. With a career largely dedicated to the advancement of best practices that leverage the PM's ability to predictably deliver business value, he has expertise in Financial Services, Telecommunications, Education, Healthcare, Transportation and Insurance. Today, Steve works with Geneca's enterprise clients to help them achieve business/IT alignment, predictability and clarity. Steve has an MS in CS from DePaul University. Contact Steve at: [steve.pieczko@geneca.com](mailto:steve.pieczko@geneca.com).

### About Geneca

Chicago-based custom software development firm, Geneca, helps its clients meet their business challenges by bringing predictability to the software development process. Getting Predictable<sup>SM</sup>, Geneca's pioneering approach to Requirements Definition and Management, has an outstanding success rate in helping its clients drive clear business alignment by identifying project objectives and success criteria. Learn more about Getting Predictable<sup>SM</sup> and Geneca's other software services at [www.geneca.com](http://www.geneca.com).

Additional copies of this whitepaper are available at [www.geneca.com](http://www.geneca.com).



Geneca Headquarters  
1815 S. Meyers Road,  
Suite 950  
Oakbrook Terrace, IL 60181

Voice: 630 599-0900  
Fax: 630 599-0908  
Toll Free: 877 436-3224  
[www.geneca.com](http://www.geneca.com)

General: [info@geneca.com](mailto:info@geneca.com)  
Sales: [sales@geneca.com](mailto:sales@geneca.com)  
Careers: [jobs@geneca.com](mailto:jobs@geneca.com)  
Press: [press@geneca.com](mailto:press@geneca.com)