Waterfall and Agile

By: Brandi Narvaez

The honor of your presence is requested at the marriage of Waterfall and Agile.

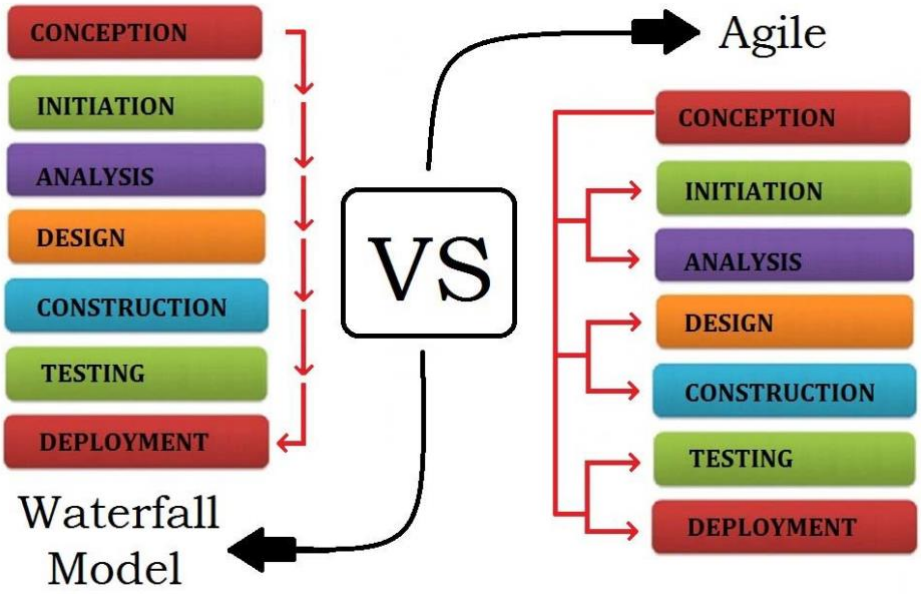
I love project management; the order, planning, controls, and a very logical beginning and end. In fact its very definition is inspirational to me, "a temporary endeavor undertaken to create a unique product, service or result" [1]. I also love and appreciate the excitement in building something new, the chaos that inevitably occurs, the solution-ing and brainstorming that happens in the process – it is exhilarating. I would assume these processes happen in all types of project management: construction, engineering, event planning, or my personal favorite – software development.

I've worked in software development since 2000 and with different groups and specialties; web-based applications using java, C++, C@, Visual Basic, you name it. I am technical enough to be dangerous but couldn't program a single line of code. I've learned enough to talk-the-talk and be an informed and effective project manager. As I've worked with and managed different software development groups, I have also been exposed to two different, but effective methodologies: Waterfall and Agile.

When I started working in software development the only methodology used at the time was Waterfall. It was fairly basic, ha a comfortable linear feel, middle-milestone based reviews & gateways, and the model was effective in producing desired outcomes. With my team, we successfully delivered many software releases using that model. The documented pros & cons of Waterfall are true with the largest risk being not able to course correct quickly and revise the 'desired outcome'. Waterfall stays true to the requirements and scope, even with a proper change management process, you can easily miss the outcome you really need in favor of what was originally documented.

At the time, a few other models were making their way into the PM realm: Rapid Application Development (RAD) and Spiral[2] – it was a way to incorporate rapid iterations on prototypes with Waterfall. It never gained the popularity that the Agile model did when it hit the scene in 2001. Agile is about being able to get feedback and respond to change (aka failing fast). It's about building products our customers want and are willing to pay for. I did not adopt Agile as a project manager. I took a few courses, earned some PDU's, and dabbled – but never, ever considered a full scale revolutionary change into my PM world. Until years later – fast forward to 2013 – I took a new role at a company that had already adopted the Agile methodology to build an enterprise software platform. The time had come for a personal revolution.

How does a strict Waterfall PM adopt Agile?



Not slowly, that is for sure. I found it to be more of a religion-based conversation; so basic as to if you believe in God or not. The Agile believers tend to be zealots and have a religious and spiritual connection to their god. I had to be willing to adopt the lingo, the process, and work within the Agile model that the team itself had fully embraced – essentially worship their god. As I tried that path, I found I wasn't getting what I needed out of the sprinting process and the desired outcomes were not what we needed as a team, a product, or a company. I was distraught that perhaps Agile methodologies were not appropriate for building and implementing an enterprise software solution. Maybe we were worshiping to the wrong god?

We were closing the projects, keeping to a true PM definition of project, but the desired outcomes were woefully lacking. We would begin a sprint and plan to build a feature without a full understanding of the requirements, no market analysis, and certainly no well thought out designs. Most of the popular Agile approaches do come with a set of prescribed roles, artifacts (aka software requirements), and ceremonies that you are supposed to follow for success. The Agile model involves failing fast, but not failing every time.

We were 'winging it' – to use a fun expression. Unfortunately the winging kept us in a losing game. We also had no warning mechanism; often we would get to a defined release date only to be told "the product isn't ready" – how did we only know that when we got to the release date? We should have known so much sooner than that and began making tradeoff decisions proactively. There are only three components of a project that can be compromised on: Scope, Resources, and Schedule – we were

always choosing "Schedule" because there was no other choice once we were at the point of late! Agile alone had failed us. While reacting to change is important, in our organization, it came at the cost of any level of predictability.

The problems seemed to boil down to the following:

1. How do I ensure we achieve desired outcomes for the product?
2. How do we make better decisions: Scope, Resources, Schedule?
3. How do we deliver on time?
4. How do we better manage & mitigate risks?
5. If we are Agile zealots why weren't we doing design/analysis before constructing?
6. We didn't have a strong and skilled product owner guiding our sprints with the voice of the customer.
7. Is the idea of planning an enterprise software release actually Agile?
8. How do I embrace the Agile model and not offend those worshiping to the Agile gods?

I attempted to solution this problem by interjecting some Waterfall. It is what I fall back to when I feel the Agile model sprinting out of control (pun intended). As a team, we agreed to do the following:

1. Define a product council – which serves as one of the feature or idea generation factories and the change management committee to approve or deny changes/requests as they enter and attempt to derail the sprints.
2. Implement a policy that nothing can be assigned to a sprint unless it has been through a preliminary process of analysis & design with our product owner and lead architect.
3. Define milestones and sequence them in a way that provides reviews, gateways, and a way to tell if we are sprinting off the track.
4. Introduce a de-risking process – which is part of the Agile religion – but one we hadn't fully worshiped as a team. Enabling us to make better decisions when it comes to Scope, Resources, and Schedules & achieve desired outcomes.
5. Core to our business strategy, we also introduced the concept of market validation before a feature is put into scope of a software release. This wasn't a change to our SDLC per say, but it made a huge difference in what the developers work on and therefore aids in ensuring desired outcomes.

Introducing Waterfall based milestones, de-risking, and adding councils and policies became a saving grace. To be fair, the Agile model doesn't shun these concepts, but doesn't regiment them either. Agile is more neutral or organic when it comes to these types of things 'working themselves out in the process.' We clearly couldn't depend any longer on things working themselves out. There are clear milestones to work towards now: those milestones are considered and incorporated in sprint planning, and are easy to identify if we have an impact to the overall project (aka

software release) due date. As a side effect, communications have dramatically improved internally and externally to our customers. One validation point – when we made the right choices applying some Waterfall methods to our Agile religion, we made our first ever on-time release date!

Interjecting Waterfall was one factor in our improvements and our ability to achieve release dates and desired outcomes. Secondarily, we also became stronger Agile zealots by adding a strong Product Owner to the team, (i.e. defined the right roles needed to be successful in Agile). We adopted the practice of analysis and design in advance of putting scope into a sprint. We meet regularly to review milestones and hold each other accountable when things are slipping, and we are able to respond to risk we see approaching and mitigate or accept impact - but this time by choice and not by default. We continue to 'fail faster' but with movement towards appropriate outcomes for our team, our products, and ultimately our customer.

We continue to grow and learn together as we work out our new Agile-fall model. I recently heard the term: "Agile-fall" and it immediately resonated with me – this is what we are doing: blending the two methodologies. We develop and test software via Agile, but release via Waterfall. In this way the developers get to bow to their Agile god, yet I can manage to a project (aka the software release) in a traditional model that the business & our customers understand: Waterfall. One of the things I like about the philosophy of Agile is its willingness to morph into whatever works best for the teams involved. It morphed to not only become predictable but to welcome the regiments we placed on it through our Waterfall experiences (much like the compromises married couples make). I had to be willing to adapt to the Agile methods and the teams had to be willing to accept some of the Waterfall process – in the end cohabitating these two methodologies has made all the difference for us and our success.

May I introduce to you the new Mr. & Mrs. Agile-Fall – may they live happily ever after.

References:

1 - A Guide to the Project Management Body of Knowledge (PMBOK Guide), Third Edition, Project Management Institute.

2 – Software Development Models: http://en.wikipedia.org/wiki/Software_development_process

3- Agile Software Development Model: http://en.wikipedia.org/wiki/Agile_software_development