Can Starting with Waterfall Lead to Better Agile? Part II: Waterfallâ€™s Contribution to Agility

Description

Introduction

In the previous article we discussed Waterfall, WaterScrumFall, big-A Agile and business agility. Dissonance abounds among organizations struggling to transition their approaches to building solutions and maintaining their existing legacy infrastructures while remaking how they evolve themselves. In attempting to navigate this they often adopt approaches that are destined ***not*** to really get them where they need to be. They need to loosen the reins on almost everything about what solutions get built, how they are built and how initiatives are managed. They have historically focused their management efforts through the lens of the triple constraint and the tautologies and assumptions on which it is based. These break down badly under the experiment-driven paradigm needed to enable agility, so a new way must be found.

One thing that has remained constant is the need to be able to assess the ongoing appropriateness of ***what*** is being implemented, ***how*** it is being implemented and ***whether*** it should continue to be implemented. In earlier times, assumptions that were revisited infrequently if at all were the basis for this, and I believe that it contributed to many a failure and a lot of wasted money, time and effort. Making a U-turn based on the answers to the ***what***, ***how***,or ***whether*** questions was extremely difficult and costly, not to mention politically fraught, so many projects meandered on, driven by inertia.

In a previous article, I discussed the danger of over-designing solutions at the outset of initiatives. It tends to lock in implementation decisions at a time when the least is known about requirements and user preferences that ultimately will be, and creates impediments to revisiting those decisions when they should be reconsidered. Once developers are let off the chain, they forcefully resist being caught, restrained and re-targeted, regardless of how appropriate and necessary it might be. Common monitoring and development metrics, such as burn rate and sprint completion percentage, and incentive systems contribute mightily to this. Companies continue to prioritize ***output*** while giving secondary consideration to ***outcomes*** as measures of productivity. ***Output*** is relatively easy to observe; there are any number of concrete events to support it, such as code modules checked in,

deployments or unit test successes achieved. ***Outcomes*** are a little more ephemeral, but ultimately they are whatâ€™s really important.

I referred to OODA in the previous article also. Accelerating this loop is critical to your managing initiatives and I assert that it will benefit you to do a number of â€œWaterfallishâ€• things prior to initiating execution of your initiatives to accomplish this.

Finally, I am involved with the Agile 2 Academy. Some of the concepts I refer to in this article come from or are based on its.

Pre-Execution

Managing initiatives requires continually answering ***what***, ***how*** or ***whether*** questions and you need a conceptual framework to do this. Pre-execution, you should establish this framework.

The first element of the conceptual framework is the **Business Case**. This defines what the solution is expected to achieve at a business levelâ€”what the strategic underpinning of the initiative is, what tactics it should enable, what assumptions are implicit in the formulation of it, what hypotheses must be confirmed or refuted to determine whether it is viable and under what circumstances it should be deemed either a success or a failure definitively.

This is hugely important. An initiative should undergo continuous scrutiny to justify the ongoing investment in it. If it reaches a point at which the justification fails, then it should be terminated and the resources it is consuming reallocated. Articulating the assumptions, hypotheses and conditions under which the initiative will be considered to have proven or disproven itself will make it easier to come to and execute a decision about its future.

The second element is **Architecture**. Architecture is a fraught term. It seems to mean something a little different to everyone. Letâ€™s define it as the front end of the design processâ€”major components of the solution with minimal focus on things like UX, UI and behavior. The architectureâ€™s use is to determine what components will be implemented, integration requirements, non-functional requirements and so on. This will serve as a basis to establish some type of envelope around the effort so that we will know whether weâ€™re setting out to build a house or an office building.

The third element is **Design.** A design defines how the architecture will be implemented. For instance, the architecture might include a shopping function with a set of catalog management, presentation and search capabilities, a recommendation engine, integration with an inventory manager and so on. The design of the solution will dictate what these pieces will look like to a user and how they will behave. It may also inform and provoke reconsideration of some architectural decisions. The design is the point at which the POs and PMs usually become more actively involved to define and qualify user stories and is one of the more traditional elements of most Agile frameworks.

It is important that stories or feature requirements can be qualified with respect to where they fit relative to the business case. Those that underlie seminal questions that will determine the viability of the initiative should be prioritized in implementation to minimize resources expended in pursuit of what might prove to be a fruitless exercise.

The fourth element is **Implementation Planning**. In traditional Agile, a major output of Design activities, as defined above, is a development backlog and that is also the case here. Although sprints and quarterly PIs may be anti-agile, work must still be prioritized, and in this step that is what happens. Initial prioritization of the work plan should be based on these few factors:

- **Decision Point Requirements:** The primary goal is to confirm or refute the initiativeâ€™s viability as quickly as possible. Implementing the components required for that should be prioritized. Facilitating rapid progression to any given state (e.g., beta, MVP, commercial release) once a productâ€™s value has been confirmed is secondary to validation.
- **Dependencies**: Given the priority of the components or stories required to meet the previous need, anything on which they might be dependent in order to function should also be prioritized.
- **PO and PM priorities**: Features and functions identified as high value by POs and PMs should be prioritized next.
- **Product Staging**: Given the priorities established in the preceding, various deliverable stages can be defined, e.g., beta, MVP, initial release, etc. These stages can be used for notional planning, scheduling and budgeting.

It should be noted that the resulting work plan is only a starting point for the initiativeâ€™s execution. Since almost any aspect of the conceptual framework might change in response to new information the plan will be fluid. Presumably, the solution architecture will remain in place; however, this too, may also be subject to revision.

The agility in Agile relates to how such changes are dealt with and allows for modifications to WHAT is being built, HOW it is being built and WHETHER it should continue to be built.

A Note on Process

To this point, it would appear that a linear process is being advocated but this is not the case. Just as development activities must allow for progress, retrenchment and realignment, so too must the Architecture and Design activities. User stories and intended functionality drive the architecture and design which, in turn, influences which features and functions will be built and how. Ideally, the architecture will remain stable over the life of the development and the operating life of the solution but change to reflect new information must always be accommodated.

Execution

Execution constitutes what most people think of when they think of Agileâ€“iterative development, sprints, program increments and so on. Agile 2 eschews prescribed execution approaches as anti-agile as they can disrupt flow and inhibit the organic collaboration and problem-solving that accelerate execution overall. Certain aspects of Agile do need to be accommodated, however:

- Continuous PO and PM participation and review of the evolving product
- DevOps, CI/CD, which will facilitate POs, PMs, Testers and users interacting with the evolving product,
- â€œProject Managementâ€• must be performed. While agile projects do not lend themselves to being managed in the traditional way, there is still a need to validate that expectations about the

delivery of major components can and will be met. If issues have arisen, it is incumbent on team leads to ensure that the appropriate people are working to resolve them.
- The team leads and initiative management team must integrate with the process of business review and confirmation/refutation. This requires collaboration, introspection and a willingness to pivot as necessary which, in many ways, relies on Agile 2 behaviors, culture and knowledge.
- Retrospectives and continuous improvement must be pursued. In classic Agile, retrospectives are conducted in conjunction with or immediately following various ceremonies, such as sprint closures. Since fixed sprints are not a feature of the Agile 2 approach these ceremonies will need to be scheduled when it is appropriate, based on the state of the development work and deliverables. Experience-based learning and continuous improvement is an important aspect of agile development maturity.

Conclusion

Running successful initiatives is a complicated business. The Waterfall approach is appropriate for a small percent of projects, specially those for which the business case, requirements, architecture and design are invariant and, therefore, the sole measure of success is execution. Itâ€™s not appropriate otherwise; however, elements of waterfallâ€™s requirements and design phases can benefit agile initiatives by contributing to the conceptual framework that will be applied to managing them. A thorough understanding of how elements of the framework connect is essential to accelerating the OODA loop and creating agility throughout the delivery team.

The Agile 2 approach to development views agility, at least within the context of the loop bound by the POs and PMs and the dev team, as resulting from participants developing their own processes in response to their particular circumstances rather than adopting predefined ones. In this article, weâ€™ve discussed a larger context in which a number of pre-execution activities can contribute to overall agility. Articulating and establishing assumptions, hypotheses, expectations about how an initiative will create value and metrics by which the progress and ongoing validity will be measured can make development work much easier, faster and responsive. Ultimately, this will set the table for initiative success.

Date Created
2022/05/12
Author
howardmwiener