HOW YOUR BUSINESS WORKS
information management
December 23, 2013

When Agile BI is Not Agile

By John Harmann

Through our collective experience at my firm, my colleagues and I have collaborated on many different projects together. And in just about every case, our differing backgrounds required us to adapt or assimilate different methodologies put forth by members of the team or our clients.

Sometimes, customers encourage us to use our experience toward determining the approach that works best; more often than not, however, customers' IT organizations want us to conform to their own methodologies. As consultants, we're trained to adapt to all types of scenarios and environments, but we've learned a lot along the way. More important, we've learned that many customers that want to use Agile methodologies don't necessarily always want to be agile, or don't fully understand the Agile approach.

What is Agile?

As its moniker suggests, Agile embraces rapid development. Agile is not a proven theorem or a finite mathematical equation with logical outcomes, like Newton's laws of motion governing gravity and inertia. Notwithstanding efforts of current string theorists who like to stir the pot and challenge anomalies to these laws, most of us will agree that Newton's theories are still pretty sacrosanct. And two plus two will always equal four. However, to be most effective, Agile must be unencumbered by restrictive, burdensome methodologies, arbitrary rules, or unnecessary processes. Clients and project leaders get into trouble when they try to tack on extra work or reporting procedures, or pin Agile down with rigid rules, rather than keeping their eye on the larger picture of project success and rapid return on their investment.

How Not to Execute an Agile Strategy

We can point to a lot of instances in which "uppercase" Agile projects can go awry. Often, customers have decided they will adopt Agile Methodology X, and then proceed to institute a vast number of rules that must be followed enterprise-wide. For example, they'll initiate a rule such as "sprints must be two weeks in duration," following it up with "you must labor through a lengthy process of distilling user features and assigning points." At the end of that two-week cycle you then are required to demonstrate something worthwhile to your non-technical business user. True Agile proponents know that trying to put these types of restrictive boxes around project progress will most likely waste valuable time and result in slower development.

In our work, we've learned to appreciate that few projects are the same. And that's okay. The spirit of Agile isn't meant to enforce rules enterprise-wide. BI projects in particular are very different than operational applications. General operational applications (and I'm sure I'm even generalizing those) seem to benefit a little more from a cookie-cutter, standardized approach. You can imagine a mobile application where you want to add a feature that allows users to tap an icon that display a customer's shipping address. It's relatively straightforward to know what the user wants, update the tables and code to provide that feature, and then demonstrate the existing app to a user.

We've found time and again that using Agile within BI projects is a different animal altogether. For instance, a user-facing report is just the tip of the iceberg when it comes to accounting for the volume of work necessary to report it. Many times, when building out a new subject area in an organization, like sales, the developer would expend significantly more effort in setting up copious volumes of data. Furthermore, end users often don't really understand what they're capable of getting out of their warehouse because they've never had the ability to slice and dice the data in that way before. Going through the extra effort to create and demonstrate a simple report that only lists customer names, just for the sake of having a front-end demo, does not make sense.

Similarly, organizations often think that Agile means that you can sprint through a marathon. They put their teams into a constant state of artificial panic and provide no time that allows developers to actually catch their breath and refine the process. Agile also allows for changing course to avoid compounding miscalculations or oversights that could jeopardize the long-term integrity of a given BI design.

Successfully Using Agile Within BI Projects

The good news is that Agile can and does work for BI in practice. To be clearer, I think Agile is more important for BI than for many other projects. Since users often don't really know what they want in BI projects, giving them the opportunity to constantly tack and adjust a project's evolutionary progress is vitally important to success. Here are a few key approaches that I've learned over the years for BI projects. Again, I don't expect them to work for every project, but they've proven to be excellent starting points for much of my work:

1. Three-week cycles work well. Depending on your team's size, it's tough to make valuable progress in two weeks and then have time to take a step back, review what happened, and plan the next cycle. Three weeks allows us to do about 2.5 weeks of work, then spend a few days on planning, high level design and retrospectives. In addition, it frees everyone up from having Fridays with tons of meetings.
2. Data are features too. One of the key assets of a BI project is the data itself — you don't have to deliver reports. A successful sprint can deliver a couple of core dimensions.
3. Plan for refactoring. DW tables don't exist in a vacuum, and often you can't uncover issues by looking at a table by itself. In a BI project, you'll often uncover many of your real data issues once you've built your complete star schema. Then you can write and perform queries to slice and dice data in different ways.
4. Know your constituency. If you have a few users who are analytically inclined and SQL savvy, by all means take advantage of it if you have time. Your sprint can end with a demo of queries and a period of data analysis. If not, perhaps have a couple of longer or internal cycles.
5. Don't forget the Retrospective. Regardless of the meeting name or approach to doing so, one of the key tenets of Agile development is refining your approach and adapting to change. That means looking at what you did, thinking about how you can improve, and continually getting better.
6. Be agile, not Agile. Being agile does not mean you have to follow exactly what another Scrum project did, or force yourself to use Kanban boards, or hold your team to official DSDM rules. Leverage the knowledge gained from prior work, but evolve to do what is best for yours.

Since the core principles of Agile involve adapting to change, stress the virtues of working closely with the customer and the team, and continually creating working software. And keep in mind that while adhering to rigid rules or dealing in absolutes definitely has merits in promoting consistent quality in many other aspects of work and industry, Agile in a BI environment functions best when it allows BI teams to advance client goals in a more flexible environment. Absolutely.

John Harmann is a founding Principal at CBIG Consulting, a professional services firm that helps clients leverage their data assets through BI and big data analytics to produce timely, effective business strategies and tactical decisions. His responsibilities include managing, developing and implementing big data analytics architectures, business intelligence/data warehousing solutions and cloud-based analytics services. CBIG's subject matter expertise extends to a variety of industries including consumer packaged goods, telecommunications, manufacturing, insurance, banking and health care. You can reach John at john.harmann@cbigconsulting.com.