

Who Doesn't Want to be Agile?

By: Steve Dine

President, Datasource Consulting, LLC

7/10/2008

Who wants to be involved in a BI project or program that is labeled slow or inflexible? While I don't believe that this is at the top of anyone's list, the reality is that this likely describes most BI implementations. This is due in large part to the waterfall approach to project management that we often employ to manage our BI projects and govern our approach to ongoing development. While BI practitioners recognize that businesses are not static and we must be able to effectively deliver value quickly and iteratively in a changing environment, we hold on to a project management approach that is neither fast nor flexible. However, applying the concepts of agile software development to BI isn't as easy or straightforward as one might be led to believe. I've been fortunate to have been involved in BI projects that have incorporated agile project management principles to varying degrees and found that some practices work better than others. The goal of this article is to provide a bit of background on agile project management & development and my experiences, both positive and negative, with applying these concepts to BI.

What is Agile?

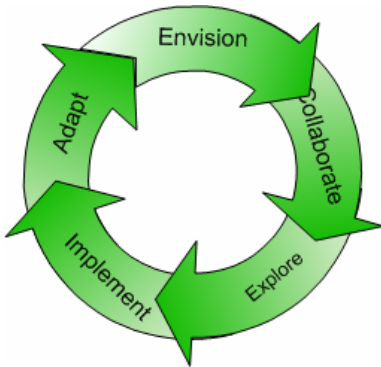
Agile software development is a set of principles and practices that was influenced by practitioners of Extreme Programming, SCRUM, DSDM, Adaptive Software Development and others. It was driven out of the need for an alternative to documentation driven, heavyweight software development processes. These principles were formally organized into the Manifesto for Agile Software Development.¹ This Manifesto stressed:

- Individuals & interactions over processes & tools
- Working products over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

The Manifesto also includes 12 principles that, among other things, stress frequent and continuous delivery, the welcoming of changes to requirements, strong face-to-face business participation, individual empowerment, sustainable development, simplicity and constant adjustment.² Given the iterative, business driven nature of successful BI projects in a quickly changing business environment, one can see why many BI practitioners quickly embrace the agile principles. However, agile development is not a process but a philosophy with a set of guiding principles. Over time, a set of practices have emerged that are based on the principles that were originally established.

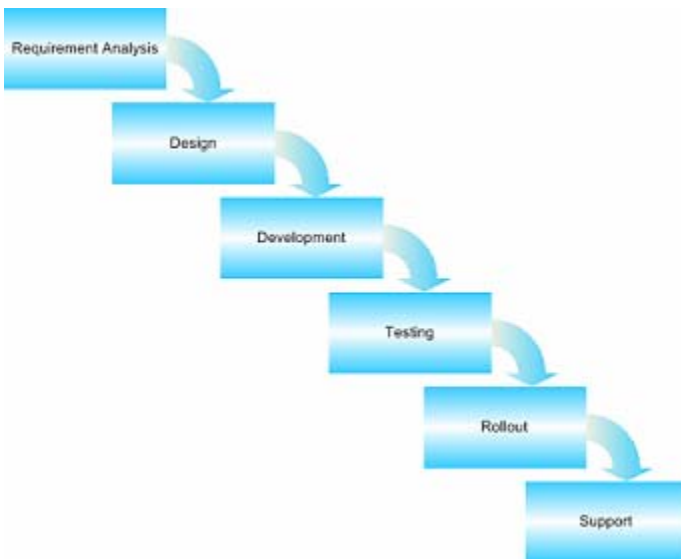
Managing a project using agile principles and practices differs from the traditional waterfall methodology in that the iterations are shorter and the detailed requirements are gathered during the collective design and development phase. Agile timelines are often measured in weeks, not months and little documentation is produced. This is achieved through practices such as storyboards, sprints, scrums, prototyping and refactoring. In agile development, testing is incorporated in the prototyping process and not left until development has been completed. Testing becomes part of the design and development process rather than its own phase. I find that this requires effective prioritization of requirements, a collaborative corporate culture, a strong project manager, low turnover in the project team and heavy involvement by the business during the design and development process.

Figure 1 – Example Agile Project Methodology



Waterfall development focuses on the phase of each discipline in a single step before moving on to the next step. One of the challenges of the traditional waterfall methodology is the long lag times between the system request and the delivery of the final product. The requirements are collected up front and given that change is constant in business, what is a requirement at the start of the project may no longer be a requirement at the end of the project. Even with delivering 90 day iterations, more than two months can pass between the time the requirements are collected and the BI system is deployed. Another challenge of utilizing the waterfall methodology for BI projects is that when it comes to BI systems, most business users tend to understand the problem, not the solution. However, the users usually aren't exposed to the solution until well beyond the design phase. This makes it difficult to effectively gather requirements and design the solution. In addition, testing is generally left to near the end of the project. As most of us have experienced, there is often little time left to make the necessary fixes without jeopardizing the delivery date.

Figure 2 – Traditional Waterfall Methodology



A View of Agile BI from the Trenches

As mentioned above, agile development itself is a philosophy with a set of guiding principles. The practices employed that adhere to the agile philosophy differ from project to project. Below are five agile development practices that I've experienced over the years in BI projects along with a subjective review of what worked well and what didn't keeping in mind that your experiences may be different.

Sprint

A sprint is a defined period, commonly two and four weeks, in which the project team commits to completing a specific set of tasks and/or deliverables. These tasks and deliverables come from the list of prioritized requirements, called the sprint backlog. The team members meet at the start of each sprint planning meeting to discuss the requirements, add the detailed tasks and estimate the hours to complete each task. The project manager ensures that the aggregate number of hours doesn't exceed the total number of hours available. At the end of each sprint, a meeting is held to review the past sprint and discuss whether any adjustments need to be made for the next one.

In my experience, sprints work well for BI projects as it keeps the business users engaged with frequent delivery of prototypes and deliverables. However, it does require some adjustment from pure software development initiatives. In many organizations, data requires integration from numerous disparate systems. Gaining access to these systems can often be difficult and be challenging to estimate. You need to consider these factors and add them to earlier sprints even though it may not yet appear on the backlog. In addition, the quality of the data can be less than desirable and cause users to lose confidence in the data warehouse early in the project. Utilizing test data rather than actual data, while potentially taking more time to generate, can often keep your project on track.

Scrum

A scrum is a short, daily project team meeting to review each team member's progress since the previous day, what they plan to do that day and communicate any blocks, or barriers that are in the way of completing their sprint tasks. There is usually a facilitator, or scrum master, that records the blocks and follow-ups on a whiteboard and is charged with keeping the meeting short and concise.

I am a strong proponent of scrum meetings as I find that it augments team communication and reduces the length of time that an issue perpetuates. There is nothing more frustrating than discovering that the data analyst has been held up for a week because they don't have access to a database driver. Scrum meetings can be especially effective with projects that utilize offshore teams that can go entire weeks without verbal communication with the onshore team. Issues are often resolved before the offshore team arrives to work the following day.

I've found that the two biggest challenges to effective scrum meetings are keeping the meeting short and securing a dedicated meeting room. When scrum meetings start lasting more than 30 minutes it seems that team members start finding excuses to skip the meetings. It is critical that the scrum master actively facilitate the meetings and that the project lead step in when the meeting starts to head off course. It is also important to designate a war room with a white board and large post it sheets. This encourages the collaboration process and reduces the overhead of having to prepare the meeting room every day.

Limited Documentation

The agile philosophy stresses communication over documentation. Many agile proponents argue that "documentation should be just barely good enough" and that "comprehensive documentation does not ensure project success but increases your chance of failure".³ I can sympathize with their view as I've experienced a number of projects where it seemed that more hours were spent on documentation than development. While these practitioners aren't necessarily advocating eliminating all documentation, they believe that the process itself should be self documenting via models and templates. Like many of the principles, the interpretation can vary widely.

The obvious benefit of reduced documentation is that more hours can be spent on design, development and testing. However, the drawbacks are that much of the interpretation of the requirements can be left to the developers and turnover in project resources can be fatal to meeting deadlines. A few years ago I was brought into an agile BI project to replace an ETL architect that had left. The team was frantically scrambling to meet their delivery date of the current sprint and the only documentation I had to review was digital pictures of whiteboard sessions, a two page project scoping document, a few report mockups and a data model. Needless to say that the process of getting up to speed on the project took longer than necessary and many of the requirements and business rules conveyed to me by other team members turned out to be incorrect.

So, what is the right amount of documentation? The answer is that it depends. It depends on the project management methodology that is employed, the size of the project team and the compliance requirements of the organization. Since I can't provide a succinct answer, below are some general guidelines that I follow:

- Utilize modeling tools to generate project documentation
- Use bulleted lists rather than paragraphs in requirements documents
- Create XLST style sheets to create documentation from XML output
- Use templates, such as code templates with header standards and interview templates
- Leverage metadata that is generated from data profiling and ETL tools
- Have a digital camera available for white boards and easels

Prototyping

It's been often said that business users understand the problem and BI practitioners understand the solution. One way to bridge that gap is to utilize prototypes. Prototyping is a strategy in system development in which a scaled down system or portion of a system is constructed in a short time, tested, and improved in several iterations. A prototype is an initial version of a system that is quickly developed to test the effectiveness of the overall design being used to solve a particular problem an iterative process.⁴ It allows you to refactor before investing significant development time in the final product.

Prototyping is a powerful practice for BI projects. I often see users struggle with identifying and articulating their requirements without any visual context of the solution. In fact, requirements often change significantly in many BI projects during testing, when this is the first time users have the opportunity to interact with the BI portal, managed reports and data. Prototyping can also save time by uncovering issues earlier in the project such as tool limitations and data availability. However, in my experience the challenge utilizing prototypes in BI projects is that it can significantly increase the amount of work for the data modeler(s) and the ETL developer(s) and they often become the bottleneck in the process. While early prototypes can be mocked up in

back office tools such as MS Excel and Visio, the next level of prototypes are inevitably created in the front-end tool sets. They rely on the data modeler and the ETL developers to create data sets to support the prototypes, whether it's sample data or actual data. When developing prototypes, you need to plan for how the data will be generated and ensure that the level of prototype coincide with the progress of the data modeling and ETL development.

Refactoring

Refactoring, in its strictest form, involves making small, continuous changes to a product's internal structure without changing its external behavior. It does not necessarily focus on adding new functionality, but enhancing its design in order to improve its adaptability.⁵ All projects employ refactoring to some extent but in a traditional waterfall approach to development, changes are usually only made in response to issues rather than enhancement of the design. One of the keys to being able to accommodate change is having a design that supports it. It is improbable to design a BI architecture up front that is impervious to change yet we resist change to our design and only focus on adding new functionality and subject areas. Over time, this creates slow and inflexible BI programs.

Two challenges to refactoring in BI projects and programs are testing and change management. Any changes to the design, however small, requires, at the very least, regression testing. This can especially be a challenge in environments with strict compliance requirements. Automating the regression testing process is critical when employing refactoring. Also, creating change levels can be helpful where each type of change is classified into three levels according to impact, such as whether the change affects the data, application configuration or both. Another way to mitigate risk with refactoring is a strong change management policy and process. Utilizing source code management tools, configuration management applications and workflow based notification and tracking systems can help audit, control and automate the change process.

Summary

BI programs and projects can implement faster and become more flexible from adopting the principles and practices that follow the agile development philosophy. However, as mentioned in the beginning of the article, applying the concepts of agile software development to BI isn't as easy or straightforward as one might be led to believe and it doesn't necessarily mean that you need to immediately toss out your project management methodology and all your development processes. I find that it is most effective when BI programs introduce agile development practices into existing methodologies in an incremental fashion. This allows program managers to discover what works best in their organization and BI program, as well as time to evaluate their ability to effectively support agile practices on a project by project basis.

¹ <http://www.agilemanifesto.org/history.html>

² *ibid*

³ Highsmith, Agile Project Management, p. 180

⁴ <http://services.eliteral.com/glossary/decision-support-systems-glossary.php>