Agile Methods and Data Warehousing

—Kent Graziano
Denver Public Schools

Introduction

Most people will agree that data warehousing and business intelligence projects take too long to deliver tangible results. Often by the time a solution is in place, the business needs have changed. With all the talk about Agile development methods and Extreme Programming, the question arises as to how these approaches can be used to deliver data warehouse and business intelligence projects faster. This article will attempt to look at some of the principles behind the Agile Manifesto and see how they might be applied in the context of a traditional data warehouse project. We will also examine a new approach called the Data Vault to see if that methodology helps. The goal is to determine a method or methods to get a more rapid (2–4 week) delivery of portions of an enterprise data warehouse architecture.

The Agile Manifesto

This is what started the Agile movement. It is a high-level statement of an approach to software development authored by a number of people in 2001. Quoting from http://agilemanifesto.org:

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

In general, most developers would agree with what is stated here (since they tend to hate documentation, structure, and project plans).

Applying the Principles Behind the Agile Manifesto

Along with the Manifesto, the authors provided a list of 12 specific principles to be followed that would define an agile process. Here is the list of the principles and a discussion of each in the context of a data warehouse project.

1. **Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.**

While every project has the goal to satisfy the customer, data warehouses have rarely been able to do it quickly. One of the first questions to address in the area is what do we mean by "valuable software" and "customer." In the data warehouse world, I equate this to a business intelligence interface (i.e., reports, dashboards, etc.) and the customer to the end user or knowledge worker.

Continued

Everyone wants the 30–90 day deliverable—max!

If that is the case, then we cannot really apply this principle to a data warehouse project until *after* the data warehouse has been designed, developed, and put into production. In order to gain some benefit from the Agile approach, I propose that in the case of data warehouse projects, we define these terms more broadly. Specifically, "customer" can be anyone from the knowledge worker to the BI programmer. In this context, if the extract, transform, and load (ETL) programmer puts into production a piece of code that populates some tables that the BI programmer needs to produce some reports, that is "valuable software." The goal then is to continuously deliver to the BI programmer populated tables that in turn can be used to produce valuable reports for the knowledge worker.

2. **Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.**

Typically, data warehouse projects hate changing requirements because it usually leads to major data model changes and a large number of ETL programming changes. To be more agile, data warehouse projects need to be architected with this principle in mind. That is they need to be flexible and adaptable. Using a normalized model as a base often helps, as does using code generators (like Oracle Warehouse Builder) to produce the ETL code. This requires that the data warehouse team be staffed with experienced professionals who have built flexible architectures in the past and who have experience with the appropriate tools. It also helps if there is an existing enterprise data model. The team must also have access to subject matter experts for the subject areas that need to be designed.

3. **Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.**

Obviously a principle we all agree on. Everyone wants the 30–90 day deliverable—max! This is achievable in a data warehouse project provided you have good scope control and approach the effort one subject area at a time. This of course begs the question "What is a subject area?" Regardless of that, we can all agree that we should not try to do the entire enterprise data warehouse (EDW) all at once.

Can we get to a point where we deliver new functionality in a couple of weeks? That will be addressed specifically later in this article.

4. **Business people and developers must work together daily throughout the project.**

Great principle—but can we get it to happen in the data warehouse world? We must at some level. Without interaction with the business users, a data warehouse project is doomed to failure. In the initial stages of analysis and design, you need access to these folks at least weekly (for interviews, design and requirements reviews, etc.). Daily interaction may be required at some times but not at others.

As stated earlier, where the Agile approach seems to really apply in data warehousing is when developing the BI reports. At this point if the data is in place, then daily interaction will insure better reports faster. Another place where it applies nicely is in the early definition of subject areas and the data model.

As always, whether this interaction is achievable may be more dependent on political factors and management priorities rather than the *desire* by the development team to be "agile."

5. **Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.**

Again, this is true for every project, data warehouse or otherwise. If you can find people who really want to be part of the project, they will move mountains to be successful. It also helps if they are eminently qualified to do the work. If not you need to get them training ASAP, then turn them loose. After the training they may need access to ongoing support or expert mentoring to get up to speed faster. Of course if you have a shortage of motivated, qualified staff, the chance of using an Agile approach successfully is pretty much zero.

If you can find people who really want to be part of the project, they will move mountains to be successful.

Once you have these people you must keep them motivated. On a large data warehouse project that really will last years (with many intermittent deliverables), this means more than throwing pizza and Mountain Dew into their cubicles a few times a month. If we can keep the units of work (i.e., deliverables) smaller, they can experience successful deployments more frequently. That is a great motivator. Everyone

Continued

The best approach I have found for this is frequent design review sessions with the internal team critiquing each other.

likes to be told "good job" on a regular basis. It also builds a culture of success.

The way to blow this is to try to deliver the entire data warehouse solution all at once—then it is a do or die outcome.

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Developers do not like to write documentation. They like to code. The pain of communication can be minimized by quick face-to-face meetings on a regular basis. This principle leads to concepts like a daily team huddle (more about this later). This can certainly be used on a data warehouse project. The more complex the project, the more important that everyone be on the same page.

7. Working software is the primary measure of progress.

The question here is what constitutes working software on a data warehouse project. Does one ETL routine count? I propose that anything we build that gets tested and moved into "production" should be considered under this principle. Hence, moving a few tables and the associated ETL code into production counts as progress (and a success). Likewise a new BI report also meets this criterion.

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

In other words, we want to avoid burnout. This is about building a software development architecture, process, and culture that make people want to be part of the team and stay motivated (see Principle #5). Given how long data warehouse projects tend to last, this is a principle worth pursuing. If you are building a true enterprise data warehouse with a Corporate Information Factory architecture, then the "project" has no end, so it is paramount that this principle be applied. This requires very good planning and scope control so there are no "all nighters" to get code into production. Smaller deliverables, delivered more frequently should help in this area. One goal then is to create the smallest valuable unit of work possible in order to keep things moving (which also keeps people motivated).

9. Continuous attention to technical excellence and good design enhances agility.

This is certainly true for a data warehouse project. A bad design or architecture will kill the project sooner or later. Either you will not be able to adapt and expand the scope of the warehouse in the future or you will find reports and data marts that you cannot easily produce. If it takes a huge effort to make a change, your project is not agile.

The best approach I have found for this is frequent design review sessions with the internal team critiquing each other. After a time, patterns of good design and bad design start to emerge and become obvious to the team. The overall competence of the team is thereby increased and as a result, the time to do design reviews is decreased.

10. Simplicity—the art of maximizing the amount of work not done—is essential.

Or stated another way—KISS (Keep it Simple Stupid!). Data warehouses are complex enough without adding work by trying to do something really slick that takes thousands of lines of code.

One of the best ways to achieve this principle is to use code generators like those provided by Oracle Designer and Oracle Warehouse Builder. While the code generated may be complex, there are no syntax errors to be fixed. In addition the tools give you visual diagrams that can be used to interact with the business user rather than trying to review data definition language (DDL) or PL/SQL with them. Often a change in requirements or design can be rapidly deployed by modifying a diagram, then regenerating the code and executing it.

Put a bunch of smart people in a room together and they will generally figure out a good solution to the problem at hand.

11. The best architectures, requirements, and designs emerge from self-organizing teams.

Put a bunch of smart people in a room together and they will generally figure out a good solution to the problem at hand. Don't label them with specific titles. In data warehousing you definitely need people with architecture, data modeling, DBA, and ETL programming skills, but that does not have to be their entire job. Allowing them to interact and help each other not only builds team moral but it

Continued

I propose we begin to think in terms of sub-subject areas.

also ensures that you have a cross-trained staff. If you are going to follow an Agile approach, you cannot afford to have only one person on the team with a specific skill set, otherwise you will experience delays when they are out sick or on vacation.

A team with the latitude to self-organize will be much more productive, once they figure out how to work together.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

At DPS we have a thing called "debate mode" where we discuss possible solutions to a problem (like what is our code promotion process). After we reach consensus we move forward but set a checkpoint in the future to revisit the solution to see if it worked as planned or if it needs to be adjusted. Over a period of time processes and procedures become much more effective and efficient, and everyone has bought into the solution. Our team has used this very effectively to continuously refine the process of moving changes to the data warehouse from development to quality assurance and then to production.

Agile Concepts and Methods

This section will examine a few Agile ideas and how they might apply to data warehousing.

FEATURE-DRIVEN DEVELOPMENT (FDD)

This is one of the many documented Agile approaches. It is based around the delivery of "features" (which usually refers to user interface features). These features are grouped into Feature Sets which represent a logical grouping of features that could be delivered to a user as a component in a short time frame. My proposition here is that a Feature Set might be equivalent to a Subject Area (or subdivision thereof) in a data warehouse project (or a single fact star schema for a data mart).

In data warehousing a "subject area" generally means a logical grouping of data elements (entities, attributes, etc.) such as data related to customers or sales. In Agile FDD a subject area means a "project" which is comprised of a group of Feature Sets. Are these equivalent concepts? Maybe. Is every Feature Set deliverable in the short term? It should be or we have missed the point.

That raises the question of how big should a subject area be in a data warehouse model? We could begin to be more agile in our approach by making the scope of a subject area much smaller than is traditionally thought of in the data warehouse world. Even though data warehouse best practices suggest we avoid the "big bang" approach to an enterprise data warehouse, many subject areas are so large that getting one done takes months or even years by itself. This is a problem.

Perhaps the answer is a new way of thinking about a subject area. I propose we begin to think in terms of sub-subject areas. This essentially translates into one main entity and the related reference data required to define its keys and attributes. Does that grouping allow us to deliver anything of value to the business, like a specific query or report? If so, then it is a good boundary for a deliverable. If not, perhaps it needs to be a bit larger and include a few more related entities.

So instead of the traditional analysis paralysis we often see, we deliver the data model, and thus the data warehouse or operational data store (ODS), in much smaller chunks. Granted, we need a conceptual model of the subject area so we have an idea of the main entities, but we deliver the subject area in increments, one major entity at a time, rather than model the details of the entire subject area before we build anything. This would allow the modelers to provide deliverables to the ETL developers in days or weeks, rather than months. If done correctly we can get a production line in place that delivers the models to the ETL programmers in a just-in-time fashion for them to populate. They in turn deliver the populated tables to the BI developers so they can build the required reports for the end users.

DOMAIN NEUTRAL COMPONENTS AND DOMAIN ARCHETYPES

FDD does include thoughts about analysis and modeling (one of the few Agile approaches that does). The concept of domain neutral components and archetypes is that there are standard models that can be used across multiple business domains. In data warehousing this might be recognized as "universal data models." This concept can

Continued

In FDD this is also referred to as the "Morning Roll Call." It is a mandatory all-hands project status meeting where outstanding tasks are listed and assigned.

definitely be used to speed things along (as proposed in FDD). At an enterprise level of data warehousing, there are commonalities that have been identified and can be reused. *The Data Model Resource Book* (by Silverston, et al.) has examples of models that apply across industries. It even includes examples of a data warehouse model and some star schemas that we all have seen used in one form or another in many places. If we, as an industry, can get over the "not invented here" syndrome and use these pre-defined models as templates, we could definitely deliver systems faster (both data warehouse and non-data warehouse related). So this is one place where the Agile world and the data world are pretty much on the same page.

TEAM HUDDLES

In FDD this is also referred to as the "Morning Roll Call." It is a mandatory all-hands project status meeting where outstanding tasks are listed and assigned. It is also where team members report completed items and ask for help on in-progress items when they need it. It is limited to no more than 30 minutes so people can get back to work. Detailed discussions must be taken offline.

This concept can definitely be applied to data warehouse projects. We are doing it at DPS. It has been very successful in helping us all to keep a handle on a very complex project and keeps everyone in sync. It provides a daily forum for positive feedback to the team and helps keep the team supervisor in the loop. It has helped foster a very positive team environment and helps support Principle #11—self-organizing teams.

EXTREME PROGRAMMING (XP)

This Agile method is based around the idea of the programmer working directly with the end user to develop an application or interface. In the data warehouse world, this approach is most likely applicable in developing a business intelligence portal or report. Assuming the data marts are already in place, a programmer using something like Oracle Discoverer, or even an Excel pivot table, can work one-on-one with a user to define and then develop the reports in a very rapid manner. This of course implies that no new data requirements emerge during the report development.

PAIR PROGRAMMING

This concept is generally associated with the XP method and entails programmers working side-by-side on one terminal. One person types while the other reviews. This actually helps them catch programming errors on the spot. Again, we have found that this technique can definitely be used on a data warehousing project. We use it with Oracle Warehouse Builder when building and deploying ETL mappings and workflow processes as well as when we input PL/SQL code into Oracle Designer. Usually we do not use it in the initial development phases but we do use it when debugging in QA. Two pairs of eyes and two brains are definitely more efficient than one in many situations.

To get to quicker deliverables we must first think in smaller increments of work.

Occasionally we even do Pair Data Modeling in Designer, with one person drawing the picture and adding attributes while the other person provides input to the design as well as reviews what has just been typed on the screen.

The Two–Week Iteration—Can We Do It?

This seems to be the goal most people think of when they talk about the Agile approach. Can this be achieved on a data warehouse project? In part, that depends on what you define as a deliverable. To get to quicker deliverables we must first think in smaller increments of work. A deliverable could be:

1. A fact table for a star schema
2. A dimension table
3. A complete star (fact and all dimensions)
4. One piece of ETL code that populates a fact table
5. A function needed by the ETL code

Continued

An Agile approach has been right in front of our faces for 10 years in the Oracle world.

6. A new report or query

Even if the original intent of Agile did not consider database and ETL type development efforts, can we not apply the principles anyway? The intent of this article is to propose some new best practices for data warehouse development. So my proposition is that we can apply the concepts and principles of the Agile approach as a means of organizing our work efforts and our teams to be more efficient and deliver something sooner rather than later. Is that not a good thing? Remember that Principle #3 indicates delivery in a couple of weeks *to a couple of months*, so a two-week interaction, while desirable, is not mandatory to be considered Agile. Perhaps we (the Oracle and data warehouse community) need to be broader in our thinking and our interpretation of Agile methods. Why can't we use Agile methods and concepts to deliver a database structure quickly? Do we have to follow the letter of the law to reap benefit from Agile thinking? I think not.

How about another perspective—who is the customer/user? In the DW/BI world my user, as a builder of data warehouse structures (ODS, data marts, etc.), is really the BI programmer. His user is the knowledge worker. He cannot deliver anything useful until he has a structure with data in it. So then delivery of working ETL code that populates that structure does put something of value in the hands of ***my*** user. Should not my success as an ETL developer be measured by my ability to correctly populate tables? In turn can the success of a data warehouse designer be measured in terms of correctly designed tables?

ORACLE CASE METHOD FAST-TRACK

An Agile approach has been right in front of our faces for 10 years in the Oracle world. The Oracle CASE Method Fast-track is a rapid application development (RAD) approach, proposed and developed by Dai Clegg and Richard Barker in 1994 (*CASE Method Fast-Track: A RAD Approach*). This was really the first "agile" methodology we saw in the Oracle world. In fact it has been around since before the Agile movement took off. According to the authors, the development of Oracle*CASE (now called Oracle Designer) with its transformers and code generators made it a feasible approach.

In summary, this method uses a combination of tight scope control, direct and continuous interaction with the users, time boxing, and iterative prototyping and builds (using the Designer code generators) to deliver a system, or sub-system in a few months. According to Richard Barker, "The aim is to build an adequate usable system quickly—not a perfect system too late for business." Sounds like an Agile method to me.

So for teams using Oracle Designer, this approach is well worth investigating if you want to be more agile in your approach.

One caution from Mr. Barker: "It requires small teams of practioners of better than average expertise, each of whom is capable of covering many aspects of the work, and close cooperation with the right users…" There's the rub—this method will not work with inexperienced developers new to Designer. Nor will it work without access to the "right" users. Again all this seems to be in keeping with the principles of the Agile Manifesto.

"It requires small teams of practioners of better than average expertise, each of whom is capable of covering many aspects of the work, and close cooperation with the right users…"

So how does this apply to a data warehouse or BI project? As stated earlier, we use Oracle Designer to design and build our data warehouse structures and to generate custom PL/SQL transformation code. The same concepts apply when using Oracle Warehouse Builder to generate ETL and process flow code. Using the web PL/SQL generator in Designer, it is even possible to quickly prototype a web-based query screen to let the user validate the data loaded in the warehouse in advance of any BI reports being developed. In the case of an ODS, those web PL/SQL modules might be the report they actually need.

Even if your team is not at the level Mr. Barker said you need to use RAD (or Agile) effectively, all is not lost. In our case, we did not start with a team of "better than average expertise." Most of the DPS team had never used Designer and they were all new to data warehousing. Because of this we did

Continued

While most Agile methods did not really have database and data warehouse projects in mind, I think it is clear that we can benefit a great deal from the concepts and principles embodied by these approaches.

not attempt an Agile approach until almost two years into the project. Through training, mentoring, and just doing the work, the team is now at a level of competence where we can use an Agile approach effectively.

Data Vault

As an aside, there is a new style of data modeling specifically designed for enterprise data warehouse design called Data Vault (see www.danlinstedt.com). This is a very flexible and extensible modeling technique that makes it possible to change and extend the data warehouse very rapidly *in small chunks*. Hence we can have faster deliverables if the project is planned correctly. We are using this technique at DPS and it has helped us to build the historical portion of our architecture in a much more incremental manner.

Conclusion

While most Agile methods did not really have database and data warehouse projects in mind, I think it is clear that we can benefit a great deal from the concepts and principles embodied by these approaches. In some cases (team huddles, pair programming) we can directly adopt their techniques. With the ever-increasing rate of change in the technology and business world, and with ever-decreasing resources, we owe it to our users, customers, and employers to examine every possible mean of becoming more effective and efficient in what we do.

With the ever-increasing rate of change in the technology and business world, and with ever-decreasing resources, we owe it to our users, customers, and employers to examine every possible mean of becoming more effective and efficient in what we do.

Can we achieve 2–4 week delivery of data warehouse components? In the case of BI reports, that should be no problem. If we cannot develop a single report in that time, then there is a serious problem with our warehouse design. As for delivering the design, tables, and the populated databases in the data warehouse or data mart, I think that it is possible to improve the rate of delivery of useful objects by adopting some of these approaches and by redefining what a deliverable is. We must be open-minded and think a little differently about what we are doing. Hopefully this article has given you some new ways to look at your approach to big data warehouse projects that will help you in achieving your goals.

About the Author

Kent Graziano is the manager for Enterprise Data Integration in the Department of Technology Services at the Denver Public Schools in Denver, Colorado. Kent is the past president of RMOUG, the past president of ODTUG, and was the first dean of the IOUG University. He has over 21 years of software and applications development experience with the last 16 years devoted to Oracle, Oracle Designer, data warehousing, and Oracle Discoverer. Kent was the recipient of the 1999 Chris Wooldridge Award (from IOUG) for outstanding contributions to the Oracle user community. In 2003 he was presented with The Doug Faughnan Award *for his dedicated service and outstanding contributions to RMOUG. He is a coauthor of* The Data Model Resource_Book *and* Oracle Designer: A Template for Developing an Enterprise Standards Document. *Kent can be reached at kent_graziano@dpsk12.org.*