Executive UPDATE
CUTTER CONSORTIUM
Access to the Experts
DATA ANALYTICS & DIGITAL TECHNOLOGIES

What's the Value of Your Data? The Agile Advantage

by Jan Paul Fillie and Werner de Jong

In a world of big data, advanced analytics, in-memory data warehousing, and real-time business intelligence (BI), the concept of a central enterprise data warehouse (EDW) appears to be something from the past. The number of sources, the level of detail, and the volumes of data to which organizations are exposed present enormous challenges to filtering out the relevant information and finding actionable insights. An integrated EDW, however, will provide an anchor point for this diversity of new data and can function as the backbone for the curation of reliable customer, operational, and financial information for both steering the company and fulfilling regulatory and compliance reporting requirements.

In this *Executive Update*, we recommend some Agile techniques and best practices that retain the benefits of a central EDW while reducing the expense and lack of responsiveness to business needs and change. This data-focused approach offers a way to define user stories in a complex EDW architecture, addressing both application and information value to users and delivering a data warehouse that provides high-quality information and is resilient to change.

Exponential Data Growth: Challenges and Opportunities

Traditionally, the data warehouse enables centralized access to data using well-defined and available sources of structured data. It is the platform for more advanced analytics and a way to store and share the outcome of that analysis. The traditional approach to developing and maintaining such a data warehouse is based on a less demanding environment yet, even then, data warehouse projects encounter significant difficulties in delivering value in time.

Taking data volumes and changes into account, it is increasingly difficult to design something of this level of complexity up front. Experience shows that, in the first releases of larger data warehouse implementations, the complete design will be revisited multiple times. In such environments it is also common to disregard data quality in an attempt to reduce complexity during the construction of the data flows. Yet the principle

of "garbage in = garbage out" often leads to limited or no data flowing into the data warehouse. In these cases, even if the design works as planned, the information value will be severely restricted. Applying Agile strategies to your EDW initiative allows for evaluation/feedback on data quality early in a project lifecycle.

Applying Agile Delivery

One thing must be understood clearly for an Agile approach to be applied to a data warehouse or any other BI development: user value is not found in the application, but rather in the information availability for reporting and analysis. Next to the availability of information, the data warehouse can provide additional value to the user by providing insight into the sourcing of data and the quality of the resulting information.

Within a data warehouse, value is often contributed to the logic in the data flow while the information output is often treated as additional functionality after the warehouse is completed. The focus at the start is often limited to getting the data in the model. For any Agile method to work in the development of a data warehouse, the main focus should be the information it provides. The first step is to use the information requirements as a driver for backlog prioritization and add the information output to the "definition of done" (the Scrum term for general acceptance criteria), as we will discuss in the user story section below. The second step is to put more emphasis on test data to support the test cases. We will cover this in more detail in our discussion of information-driven development.

On top of the requirements of any Agile development, an EDW Agile team could benefit from the use of the following additional elements:

- A metadata framework (to provide traceability from sources to data warehouse output and to handle data processes)
- A way to define user stories that allows for completion of functionality in iterations
- Incremental insight into data quality, supporting the iterations
- Greater focus on the creation of test data in test-driven development and a move toward information-based continuous integration

The Metadata Framework

Right at the start of development, the Agile team should have a metadata framework. If this framework is not available, the first iterations should focus on creating it. The metadata framework involves more than just a repository where technical and business metadata is stored. It is tightly linked to the data warehouse model and allows traceability to all data residing in the model.

When selecting or designing a metadata framework, it is important to realize that some aspects are more important than others. Basically, the framework must allow you to administer and track business- and

technical-related items involving the warehouse. In connection with Agile development, the framework must be able to cater for process definitions as well as available functionality. The process definitions typically contain base runtime parameters as well as source and target definitions. Metadata will then allow for a quick impact analysis per iteration, increasing productivity and reusability.

Other typical items the metadata framework should provide are lineage information related to import and export of data, transformations performed on the imported data, functional and technical timelines, detailed runtime information, and parameters used on runtime and exception logs. The framework can be easily extended with data quality metrics or orchestration of data flows, but those can be considered nice-to-haves, as data quality should be addressed actively and not in a passive reporting style.

Defining User Stories

In an Agile approach, the desired functionality is generally described in a story-like fashion. Starting with a general theme, this gets broken down into one or more "epics." When even more information is known, the end-user requirements are further detailed in user stories. Most Agile practitioners would consider recognizable output like reports as the user stories in a data warehouse environment. But if you look at the required information, any report will require multiple data flows that cross all the layers of a data warehouse (see Figure 1). Reports are only the tip of the iceberg. For a report to be made accessible in the user access area, at least one data mart must be loaded with dimensions and facts. In turn, the data mart summarizes or combines facts from the data warehouse layer. The dimensions are copied or summarized from the data warehouse to the data mart as well. All the facts and dimensions in the data warehouse layer get their data from one or more sources. These internal or external sources are loaded to the staging area at the start. For short iterations or sprints, the user story would not fit in the sprint duration (two to four weeks) and therefore not deliver a working solution at the end of the sprint.

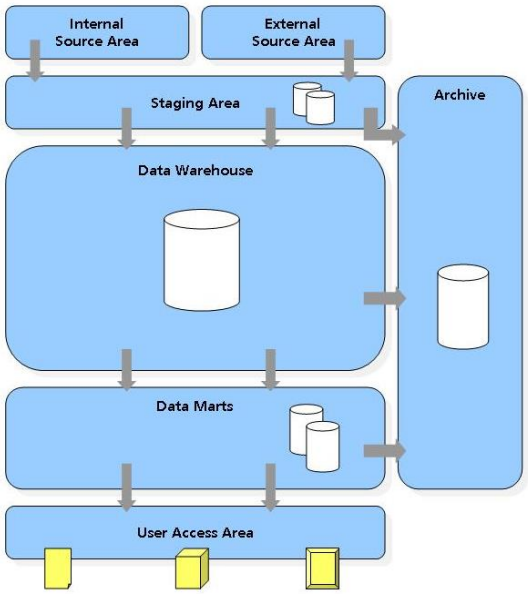

Figure 1 — Layers of a data warehouse architecture.

The way forward is to try to break a report definition down into multiple user stories. In practice, this works best when applying the perspective of dimensional modeling. Any report will provide one or more measurements or facts against one or more dimensions. The following example shows how this can be done:

> *In a supply chain organization, an Agile team was developing a new data warehouse. The first themes focused on subjects such as sales and shipments. Breaking the sales theme up into epics, the team discovered facts such as number of articles, sales amounts, and number of customers. To be able to view these facts in reports, the team found several dimensions and added them to the list of epics. Most important seemed to be customer and product. Looking further in the backlog, the team found that a lot of other themes and epics were dependent on these dimensions. The product owner therefore decided to move these items up in the backlog.*

In most cases, the customer epic and its corresponding entity will be the central and most important subject in a data warehouse. Thus, it would be advisable for any Agile EDW endeavor to target this within the first iterations. To return to our example:

> *After selecting the customer epic to be evaluated further, the team needed to break the theme down into user stories. They tried to estimate the development of the customer epic as a single story but failed because of the diversity of requests. These ranged from wanting to know the top 10 customers per product segment to customer groups by spending, customers by geographies, and hierarchies of customers based on various profiles. The first user story selected was much more basic: "I want to be able to identify my customer by name, so I can look at his/her orders." The development team then estimated the user story again and moved it to the sprint backlog.*

Depending on the complexity of the customer data and the number of sources that will provide customer data to the data warehouse, the Agile team may want to consider splitting the user story up into "customer from source x," as this will allow completion within the iteration. The goal must be to fit two or more user stories from source to report (if report is the chosen front end of the data warehouse) inside the time frame of a single sprint. One approach is to split user stories into front-end and back-end stories. The front-end stories contain end-user functionality like reporting, while the back-end stories focus on providing source data to the data model.

Because the value of the back-end user stories can be hard to define from a user standpoint, take a smaller part of functionality instead and have something the users can review from source data to provide useful information. The resulting "slice" of functionality can be designed, built, and tested within the time frame of a sprint. This will provide the required data flows and the new information/insight to the end user, making it possible for him or her to review both using the "definition of done." In incremental delivery of these slices, the result will be in the full definition of each dimension and fact in the data warehouse central layer. Following sprints can focus on just the output depending on the priorities of the product owner.

Incremental Insight into Data Quality

Poor data quality has a high impact on EDW implementations and is probably the primary cause for the failure of such initiatives. Either poor source quality will result in poor output quality, or the EDW project will have to resolve the source quality problems, resulting in an increased design and development effort. For a traditional waterfall project, this will decrease the business value of the result and/or produce enormous scope creep and multiple change requests. Consider the following example of the garbage in = garbage out phenomenon:

> *A data warehouse was being developed for a large insurance company. As the data quality was assumed to be sufficient, the development team designed and developed the data warehouse to handle any source data that passed the basic quality tests of the staging area. The initial development was completed in a year, after which the system passed both unit testing and system testing. Finally, for the acceptance test, a data set was loaded with production-like data. The end result was that no data could be shown in any of the reports. Only a small portion of the source data passed through the staging tests, and not all expected data could be delivered by the source. In a subsequent release, the entire design was revisited.*

To tackle data quality problems before they arise, it may be prudent to perform data profiling analysis on the source data. Before any source is taken into account, data is analyzed for its ability to support the required information output. Besides data profiling, further analysis on duplication and/or specific aspects of data quality can be required for particular information needs. For example:

> *A master data management solution was being implemented for a global manufacturer of medical devices. Prior to this project, a large cleansing effort had been undertaken on the source systems. Based on this effort, the client assumed the customer source data to be of high quality. To guide the definition phase, the implementation team proposed a proof of concept showing the information benefits of the master data management solution. In the first profiling of data, the team found numerous missing values, non-unique key attributes, and several other deviations from agreed standards. Further analysis showed that information was hidden in fields that were not intended for this purpose and that the duplication percentage was over 10% of the customer data. In more than 40% of the analyzed records, the team discovered one or more deviations from data quality rules.*

This second scenario would have been an utter disaster if the team had not conducted its proof of concept; implementation would have been well under way before the data issues were detected, and a large amount of funds and resources wasted. Thanks to the proactive data profiling effort, the project was cancelled relatively early, preserving relations with the customer and keeping the budget impact to a minimum. New projects were undertaken with the preserved funds to deal with the data quality. If the team had taken an Agile approach in the second scenario, the results of quality analysis would have led to better understanding, and the team might even have been able to deal with the issues in time instead of cancelling the project.

Data warehouse development has always dealt with the issue of data quality in three distinct ways:

1. **Ignoring data quality issues and continuing development of the application in the best way possible.** This approach generally results in heated discussions with end users when they are confronted with the output of the data warehouse.

2. **Highlighting the principle of garbage in = garbage out.** This occasionally brings some awareness to the end users, but since they typically consider the source data fit for purpose, the problem is usually underestimated.

3. **Up-front data investigation.** This approach will reveal major flaws in the data, but it takes quite some effort and delays before any development can be started. Secondly, the investigation can only be performed on a general level because the specific use of the data is rarely fully understood.

All the approaches listed above are flawed in the sense that, even if insight into the quality is gained, information value will not increase. Additionally, problems caused by bad quality are not always obvious during development. Developers are often not aware of what information the system should produce. An Agile approach has an advantage here.

Information-Driven Development

Organizations should embrace a test (data)-driven development strategy based upon insight from the source data targeted at delivering information to the end users. This ensures you are working as effectively as possible and getting the best out of the system. Developers often have a nagging feeling that something is not going to work, but they lack the tools or time to explore the reason. Creating scenarios of what the system will have to endure empowers the team to face difficult challenges and increases the overall understanding of the system. Knowing in advance what the result of the system should be and working toward that not only increases confidence, but also makes the development job much easier. Another benefit is that this approach lowers any barriers between team members and signals undocumented assumptions. In an EDW, this requires selecting or creating test data to support the expected information output of the system.

Information-driven development is tightly linked to the necessity of having insight into the data quality and utilizing test (data)-driven development. Understanding what data resides in the source(s) will help the team establish scenarios. It enhances the quality of the system, as designs will become more complete, and defensive programming will become standard practice.

In extension of test (data)-driven development, ensuring that the system can grow per iteration requires a different approach to the normal development of a data warehouse project. Typically for a data warehouse implementation, the number of processes is huge, and often a small change has a great impact. To ensure that existing code is still working and provides the same information, daily runs of the entire system need to be performed. These runs of the data warehouse will load predefined test data using the latest code the

team has checked in. The information output is compared to the expected results and failed or approved. As Figure 1 shows, the dependency of data flows between the layers of the data warehouse requires specific attention to upstream failures.

Vision

The complexity of data warehouses is increasing, largely due to changing business dynamics and the increase in volumes and sources of data. This makes data organization and restructuring efforts hazardous undertakings. Data curation of large data stores has yielded valuable new insights, and thus this capability is proving to be an important asset for innovation. Yet estimations suggest that the volume exponentially increases year over year, a condition that prevents big data analysis from being a fixed part of the BI support system.

The added value of the EDW remains its reliability in providing high-quality information to decision makers in day-to-day operations. The effort of creating such a system and keeping it up to date and aligned with the requirements of the business is proving difficult. Traditional ways have shown to be too much focused on the data processes and too little on the actual value in the information created. End users often face the difficult task of validating a data warehouse design without being shown the information the system will create. This information aspect is strangely disregarded in most traditional data warehouse projects.

Companies are harnessing the latest trends in technology to extract value from their data. The driving principle of EDW development should be the timely delivery of important business value, which primarily derives not from the application, but from the information output and storage. We recommend that you take a practical approach and consider adopting Agile strategies to provide flexibility and insight into data quality at every step of the way.

About the Authors

Jan-Paul Fillié is an experienced strategy consultant and Certified ScrumMaster at IBM, specializing information architecture, analytics, and cognitive solutions. His main focus is improving data-related project delivery through Agile and iterative processes. Mr. Fillié frequently publishes on business intelligence-related subjects in books and articles and provides training to colleagues and clients. He can be reached at Jan-Paul.Fillie@nl.ibm.com.

Werner de Jong is an experienced information architect at IBM, specializing in business intelligence, master data management, and metadata-driven warehouses. He has a particular knack for testing. Mr. de Jong frequently provides training to colleagues, clients, and students. He can be reached at Werner.de.jong@nl.ibm.com.