

Exclusively for TDWI Members

Business Intelligence

THE LEADING PUBLICATION FOR BUSINESS INTELLIGENCE AND DATA WAREHOUSING PROFESSIONALS

JOURNAL

AGILE DW

Agile Data Warehousing with Integrated Sandboxing



Stephen Brobst is chief technology officer for Teradata.
stephen.brobst@teradata.com



Michael McIntire is a principal architect for eBay.
michael.mcintire@ebay.com



Edward Rado is a senior technology manager for WellPoint.
edward.rado@wellpoint.com

Stephen Brobst, Michael McIntire, and Edward Rado

Abstract

The demand for greater agility in data warehouse implementations is emerging as a sure sign of maturity in the industry. With methodologies and best practices that are now well established and generally agreed upon by industry experts, we have sometimes forgotten how to act with the urgency that is reminiscent of earlier days. The philosophy of the agile data warehousing techniques described in this article is borrowed from the corresponding deployment of agile software development in organizations that have been so successful over the past several years. Adjustments are made to accommodate the enterprise nature of data warehouse deployments.

Introduction

The Manifesto for Agile Software Development (Beck, et al, 2001) puts forth these principles:

- Value individuals and interactions over processes and tools
- Value working software over comprehensive documentation
- Value customer collaboration over contract negotiation
- Value response to change over following a plan

The agile development methodology recognizes the value of processes, tools, and plans, but individuals, software, and collaboration are valued even more highly. The underlying philosophy that drives these values in data warehousing puts the highest priority on satisfying end-user (knowledge-worker) requirements through early and continuous delivery of analytic capability. This does not mean that requirements documents, design documents, entity-relationship diagrams, and data dictionaries are not important—it simply puts a greater emphasis on delivery than on process.

The agile philosophy takes the position that if a project is going to fail, it is better to know after one month than after 15 months.

This will make traditionalists nervous and even uncomfortable. The agile methodology rejects the artifact-driven approach managers have grown accustomed to over decades of software project management. In many cases, artifacts are created as “contracts” to control change. Artifacts provide specifications that are stable enough for implementation to take place without in-flight changes; such changes pose problems for developers who find it difficult to deal with deliverables with “moving targets.”

In contrast, the philosophy of agile development wholeheartedly embraces change. The agile manifesto takes the position that facilitating change is more effective than attempting to prevent it. This observation is particularly relevant in the world of data warehousing, where meaningful artifacts (in the form of traditional requirements documents) are elusive.

A typical knowledge worker can tell you the first question he or she wants to ask, but cannot specify the second question until the first one has been answered. This situation is quite frustrating for traditional project managers who want to nail down requirements as a formal specification of project deliverables. The reality in data warehousing is that such specifications are nearly impossible to create.

In data warehousing, the primary goal is to provide access to information that has high value for decision making. However, it is hard to know what information will have value before actually working with it. An underlying theme of the approach we describe is the need to expose and demonstrate the value of new data quickly and inexpensively while allowing organizations to “disprove” value in the same fashion. The agile philosophy takes the position that if a project is going to fail, it is better to know after one month than after 15 months (Jeffries, Anderson, and Hendrickson, 2000).

The intent of agile data warehousing is to create a culture of innovation wherein data exploration and construction of new analytic applications (leveraging existing data in the warehouse) is encouraged without the overhead of long project cycles and painful financial justifications. Small budgets are allocated for experimental use of data in the data warehouse; a “sandbox” environment is created to make it easy to extend the data warehouse and assess the value of new capabilities. This approach makes it cost effective to encourage accelerated innovation within the enterprise data warehouse.

Traditional Implementation of New Data Warehouse Capabilities

The rigor of best-practices methodologies has its own set of pitfalls when you are trying to foster innovation within an organization. The overhead inevitably associated with a disciplined approach to project implementation starts with getting the sign-off to begin a project in the first place. Business-case analysis,

return-on-investment discussions, and politics all provide hurdles to initial implementation. Information discovery with detailed requirements gathering, logical data modeling to extend the data warehouse design, and data mapping from source to data warehouse target tables can also hinder a project's beginning.

We begin implementation with negotiation of service-level agreements for the new capability, capacity planning to extend the data warehouse platform (if necessary), ETL (extract, transform, load) design and construction, initial data loading, analytic application construction (with many steps buried in this task), and so on.

Of course, once extensions to the data warehouse are in place, there will be a brand new set of tasks related to putting the new capability into production. Formalized unit, system-integration, performance, and user-acceptance testing must be executed with appropriate sign-offs, including data quality certification. Backup and recovery procedures must be extended to the new data and application components of the environment. Production documentation, monitoring procedures, and performance management will need to be put into place. In addition, associate training (both operations staff and knowledge workers) are required for the ultimate success of the new capability.

While most would argue that all of these steps are essential to a well-managed project deployment within a mature data warehouse environment, it is difficult to do anything in less than 90 days within this model, and sometimes 90 days is just too long! Imagine that marketing has an outside source of data that it wants to integrate into the data warehouse, but is not yet sure if the data has high value. In this scenario there is a clear need to experiment with the data before signing a purchase contract. However, the full project lifecycle methodology associated with the best practices we've described is not appropriate for bringing in the external data to enable such testing. There is too much

overhead for what might be a very short-term data requirement. The success of the "prove value fast or fail fast" approach requires that projects be measured in days or weeks, not months.

Integrated Sandbox Implementation

The goal of an integrated sandbox implementation is to provide a method of getting data into the data warehouse without the overhead of a full-blown development methodology. The strategy is to allow load-and-go analytics to enable fast implementation of non-certified content into a sandbox area that can be used in cooperation with (certified) content in the enterprise data warehouse.

The success of the "prove value fast or fail fast" approach requires that projects be measured in days or weeks, not months.

The sandbox is a special database area set aside for use by knowledge workers without the formal controls instituted for the production data warehouse tables. Knowledge workers are allowed to load (and drop) their own tables, and are given significant latitude in managing the sandbox database. Space allocation (within defined capacity constraints), content, and loading frequency are managed by the knowledge-worker community with minimal support (or interference) from production DBA staff. In some cases, separate sandboxes are allocated in alignment with distinct knowledge worker communities (e.g., marketing versus finance) to minimize cross-organizational dependencies that increase project overhead.

The sandbox is typically used to load raw data with minimal conditioning and cleansing (as these steps

are quite expensive in a traditional implementation). The use of data in this form increases the burden for writing SQL because workarounds are often necessary to obtain meaningful results. It would not be unusual to see significant gymnastics in the SQL code for JOIN specifications and WHERE-clause predicates to access data in a sandbox. However, the cost of this extra complexity in the SQL code is more than made up for by the expedient availability of the data that is attained using the load-and-go approach.

Replicating a full copy of the production data in a test environment—and keeping it up to date—are not achieved without a significant price tag.

Since the sandbox is meant to provide a proof-of-concept environment rather than a hardened production environment, little effort is merited for performance tuning. Use of advanced indexing, SQL tuning, and other database functions is unnecessary and is usually deferred to full production implementation when the sandbox content is promoted into the production data warehouse. High-powered parallel processing can deliver the performance required for sandbox analysis without significant additional effort from, or the expertise of, knowledge workers.

We recommend deploying the sandbox on the same platform as the production data warehouse because obtaining value from data newly introduced for analytics will usually require access to existing production data as well. It will be important to be able to efficiently join and pursue data exploitation across the production data and the “experimental” data in a rea-

sonably integrated environment (without the overhead of full integration implied by a single, unified data model). After all, it is the integration of data and relationships across multiple subject areas that provides the analytic value in a mature warehouse environment. An added advantage of our recommended approach is that if it later becomes appropriate to fully integrate the experimental data into the production data warehouse, it is much easier to do so if the experimental data already coexists with the production data on the same platform.

There are two other possible deployments: (1) the sandbox coexists with a shadow copy of the production data in a test environment, or (2) the sandbox is kept on a platform separate from the production system and federation software is used to integrate at the middle-ware level.

The problem with the first option is cost. Replicating a full copy of the production data in a test environment—and keeping it up to date—are not achieved without a significant price tag. Most test/development environments use only a sample of the production data, or limited historical retention, to manage cost. However, lack of access to the full production data often makes it difficult to achieve proof-of-concept results; the sample may not align to the requirements of the analytic application, or the lack of historical retention may not be acceptable. These complications add to the proof-of-concept’s duration, because the test/development environment must be repopulated in accordance with application requirements (and multiple applications undergoing fast-track implementation will typically have distinct data-content requirements).

The problem with the second alternative is performance. Sandboxes leverage data that is already in the data warehouse with incrementally acquired data. Deploying the sandbox on a separate database on a separate platform (even if using the same database technology) will require joins across a network. This

technique is supported by federation software, which is available from several vendors. A federated software implementation involves retrieving data from two or more database platforms (which may or may not use the same technology) and integrating the data at the middleware layer (typically executing on its own platform). While federation has proven to be extremely useful for integrating small volumes of data in support of OLTP applications, its use in high-volume data warehouse deployments has not been as successful. Joining two or more large tables using middleware that is orders of magnitude less efficient than a parallel RDBMS results in unacceptable performance, even for proof-of-concept deployments.

Integrating a sandbox into the production database platform delivers maximum performance and ease of use when using the experimental data in cooperation with the production data in the warehouse. The main objection to integrating the sandbox database with the production database onto a single platform is the loss of control in the production environment. The idea of end users (knowledge workers) managing their own space and having permission to create, modify, or delete tables, as well as having application development privileges, on the production system is horrifying to many DBAs. However, allowing knowledge workers some degree of freedom in the sandbox area does not imply a free-for-all on the production system. Well-thought-out governance is required to ensure success of the sandbox environment.

Governance

The purpose of the sandbox is to give a limited number of users quick access to data that has not yet been provisioned into the data warehouse. This data is for limited use in proof-of-concept projects. The approach is to use lightweight methodologies consistent with agile software delivery for acquisition of data for the purposes of developing and quickly validating new analytic processes. We recommend deployment of the project's experimental data on the production

platform for effective use in conjunction with content from the enterprise data warehouse. However, deployment in a production platform increases the need for managing resources consumed by users of the sandbox so as to avoid a negative impact other workloads.

While federation has proven to be extremely useful for integrating small volumes of data in support of OLTP applications, its use in high-volume data warehouse deployments has not been as successful.

An agile development principle is to encourage business people and developers to work together throughout a project. An effective way to make this happen is to involve business knowledge workers directly in the development process and to give them the ability to create and load their own tables in the sandbox area. However, this is not meant to be a recommendation for complete anarchy in the data warehouse environment. A space allocation limit should be set. It is up to the knowledge worker community to internally self-manage their storage use so they stay within the storage allocated. In line with the principles of agile development, we rely on motivated individuals and trust them to get the job done rather than imposing burdensome processes that slow down delivery.

The sandbox area is a development environment designed to facilitate direct cooperation between the business and the data warehouse team. This works

best when there is intensive, bidirectional interaction between knowledge workers from the business and DBAs from the data warehouse team. Education will take place on both sides of the fence when collaborative development is working well. The DBA gets intensive exposure to the business requirements, and the knowledge workers participate directly in technical design processes.

Data in the sandbox area should have a limited lifetime—long enough to prove its value but not so long as to become entrenched as a permanent resident.

However, it is critical to recognize that the sandbox area is not meant to be a production deployment environment. As such, the number of users and the scope of the work performed within the sandbox should be limited. Workload limits with automated resource governors should be put in place to ensure that runaway queries do not impact production activities.

Governance must be used to prevent “shadow” production jobs from taking hold within the sandbox area. Data in the sandbox area should have a limited lifetime—long enough to prove its value but not so long as to become entrenched as a permanent resident. A typical (allowable) residency is about 90 days. Scripts should be executed on a regular basis to detect sandbox tables that have existed for extended periods and to flag them for either retirement or promotion into the production environment.

To preserve the flexibility and lightweight nature of the sandbox area, production procedures for such tasks as backups, data loading, and performance

monitoring are normally deferred until after the new data’s business value has been proven. This avoids the overhead of operational documentation, service-level agreements, and job scheduling. These tasks will become important once the data value is proven, but our proposed methodology must avoid front-loading data-acquisition initiatives with excessive process and development overhead until a business-value proposition for such costs is justified. The benefit associated with automating these tasks also provides incentive for promoting tables from the sandbox into the integrated data warehouse environment.

The agility embodied into the sandbox area typically means that there is less rigor in terms of data quality controls, auditability, and security management when compared to the production data warehouse environment. As a result, governance is required for content and outputs from the sandbox area to ensure that the integrity of the overall analytic environment is maintained. Privacy protection, of course, is a major concern of any data warehouse environment. If there is any sensitive data such as personal health information (PHI), credit card numbers, Social Security numbers, and the like, then we recommend that separate (private) sandbox areas be deployed for only those users with data access rights to avoid unauthorized access.

In addition, certain controls will be appropriate for the sandbox’s output. The intent of the sandbox is to facilitate quick delivery of value from new subject areas of data with integration capabilities into the production data warehouse environment. However, the sandbox data will not have been taken through the same level of rigor (in terms of data quality certification and audit controls) as the production data in the warehouse. Thus, restrictions should be put into place to prevent sandbox data from being used for financial reporting purposes.

The liability associated with financial reporting makes it unacceptable to drive such reports from the sandbox. While quick development of capabilities and feasibility

testing is encouraged for integrating new financial data into the data warehouse, no reports or content to be used for internal decision making or external reporting should be allowed from the sandbox. Promotion of the data from the sandbox into the production data warehouse environment, with the appropriate quality certification and production audit controls, should be required before allowing financial reports to be based on the newly acquired data.

Promotion into the Enterprise Data Warehouse

The agile framework proposed for acquiring new data into a data warehouse encourages rapid delivery and attempts to avoid front-loading delivery projects with creating artifacts for requirements and design. The measure of success in this methodology is whether data is made available to knowledge workers quickly and efficiently. In this way, knowledge workers are better able to define and refine requirements through actual use of data rather than by reviewing documentation. Review of documentation, diagrams, and similar material is ineffective because of the abstract nature of these artifacts (Hunt and Thomas, 1999). There is no substitute for putting knowledge workers in direct contact with the data upon which they'll make their business decisions.

On the other hand, these artifacts do add value to the long-term sustainability and maintainability of the data warehouse environment. The key is to deliver access to the data to knowledge workers with highest priority and use their feedback to drive design tasks (such as logical, physical, and semantic data modeling). Rather than treating these design tasks as a prerequisite to getting data into the hands of the knowledge worker, agile methodology reverses the paradigm. Getting data into the hands of the knowledge worker is a prerequisite to facilitating the design process.

The development team explicitly includes knowledge workers as key participants. Promotion of data from the sandbox into the production data warehouse environment is predicated on proving the value of

the data as the foremost requirement. Once this has been achieved, a team composed of both knowledge workers and IT staff embarks upon an iterative process for designing the logical, physical, and semantic data models for deployment.

Although it will make sense to promote decision-making processes and data content from the sandbox into the production data warehouse environment (when ongoing value has been demonstrated through its use in the sandbox) in many cases, sometimes it will not make sense to pursue promotion when expected value cannot be demonstrated or when use of the data is for a one-off project without ongoing value. Both are good outcomes from the process because we are able to either succeed or fail quickly.

The intent of the sandbox is to facilitate quick delivery of value from new subject areas of data with integration capabilities into the production data warehouse environment.

There are many cases where one-off analysis brings tremendous value to an enterprise, even though it will not make sense to promote data content into the production data warehouse. For example, in performing analysis to assess the business implications of an acquisition, it is useful to quickly bring data into a sandbox to perform analysis related to overlapping customer populations, channel utilization, and risk profiling. However, it usually makes little sense to promote such data into the data warehouse unless the acquisition actually takes place.

Similarly, there are many kinds of “one-time” compliance and regulatory requests that must be satisfied with data that may not be in the production data warehouse, and yet it may not make sense to integrate this data if there are no expectations of its use for future analysis.

A key tenet of the agile methodology is to avoid making investments in features, functions, and data that are not valuable today. Future promise of use does not justify the distraction and resources expended for capabilities that will probably change before they are exploited. This is often referred to as the YAGNI (you aren't going to need it) principle in agile development communities. The value is in simplicity of design and development rather than over-engineering for future and elusive requirements. Iterative development and delivery is fundamental to this approach.

There is no substitute for putting knowledge workers in direct contact with the data upon which they'll make their business decisions.

In cases where promotion to production data warehouse tables is merited, the agile methodology calls for frequent (typically daily) cooperation between the data warehouse developers and knowledge workers. With the benefit of hands-on access to data, knowledge workers can directly participate in design decisions for the logical, physical, and semantic extensions of the data warehouse to accommodate the new data. Similarly, this hands-on access to data in the sandbox facilitates rapid delivery of business rules for translating raw source system data into the target tables that will be integrated into the production data warehouse.

While agile methodologies place greater emphasis on people and communication than on tools and processes, we find that the use of tools for capturing design metadata (for the data models) and transformation metadata (for mapping from source data files to target data warehouse tables and driving the ETL programs) is essential to rapid delivery. Testing and continuous integration is made much more efficient with metadata-driven approaches to design and deployment. Ideally, during the development process there is a working system for data acquisition from source to target (at least) every night. Daily “scrums” (meetings) are held to identify incremental accomplishments, immediate tasks to be completed, and any obstacles that need to be removed (Murphy, 2004).

The promotion of content and decision-making processes from the sandbox to the production data warehouse provides significant benefits in data value. Logical data-model extensions to the data warehouse will ensure proper integration of content in a way that facilitates ongoing maintainability and flexibility. Physical data-model extensions preserve the relationships embodied in the logical data model while organizing database structures for optimal query performance, storage utilization, and loading efficiency. Semantic data-model extensions provide usability of the new data content in cooperation with existing data in the warehouse.

Note that data models may change dramatically when promoting content and decision-making processes from the sandbox. It is often appropriate to generalize the sandbox design for integration into the production data warehouse. This is part of the natural evolution of software development that is encouraged by agile methodologies. Rather than designing for the completely general case up front, the developer assumes that requirements will surface and be better understood by collaborating with the knowledge workers in the sandbox area. Extensions and iterative design are

best deployed with the expectation of evolution rather than the assumption that change will not occur.

As the design stabilizes for release into the production environment, adopting more rigorous controls will add value. The automation and production management of jobs for extraction, transformation, and loading from source systems to the data warehouse provides reliability in acquiring up-to-date data for knowledge workers. Making this a production task also implies a measure of data-quality certification and auditability associated with the data content.

In addition to data content, any analytic applications and decision-making processes developed in the sandbox area must be migrated into the production environment. This will usually involve the mapping of analytics through semantic data-model extensions rather than by directly accessing raw data tables, as would likely have been the case in the sandbox. However, the semantic data model extensions will have been designed with direct input from the knowledge workers based on actual use cases developed in the sandbox.

Our experience is that the execution of each of these tasks is much more effective when there is direct involvement and collaboration with the knowledge workers who will ultimately be the users of the end product (data) in the data warehouse. Thus, documentation and the creation of other artifacts focuses on what is required to successfully operate and maintain the data warehouse—rather than on requirements, per se.

Conclusions

There are many variations on the agile deployment of software solutions: Extreme Programming (XP), scrum, agile modeling, adaptive software development (ASD), Crystal Clear, dynamic systems development model (DSDM), feature-driven development (FDD), lean software development, and Agile Unified Process

(AUP), among others. No single methodology is appropriate in all cases. In this article we have borrowed from these methods and suggested an approach using a sandbox area integrated with a production data warehouse environment to facilitate rapid delivery of value from new data content.

The execution of each of these tasks is much more effective when there is direct involvement and collaboration with the knowledge workers who will ultimately be the users of the end product.

From our experience we know that this approach is useful in quickly delivering a business case for including (or excluding) new data content into a warehouse without prolonged investment in front-loaded design and documentation tasks. This approach is also valuable for one-off projects where full investment in a traditional lifecycle model of delivery is not appropriate. However, we also believe that promotion of data content from the sandbox into the production data warehouse is critical to obtaining maximum value from new subject areas of data.

One of the core advantages of this approach is that it provides an incremental method of delivering new content to knowledge workers without encouraging the proliferation of data marts. By providing a direct path of promotion from the sandbox to the enterprise data warehouse, new subject areas of data become available for (re)use throughout the enterprise. Furthermore, the approach makes it possible to experiment with

new data and new decision-making processes without undue overhead. This ability to fail fast and cheap or prove success quickly and efficiently will encourage innovation, which will yield competitive advantage.

Providing knowledge workers with quick access to data for analysis as a repeatable service makes an enterprise more agile. Facilitating faster deployment of processes that promote informed decision making can bring competitive advantages to an enterprise. The challenge for BI and DW implementation teams is to create a working framework in which the processes and procedures for governance of these activities are in place and known by all the parties involved, from provisioning the physical structures to knowing how to make the decision.

We believe that agile data warehousing with production sandboxing significantly increases an organization's capability to more quickly deliver great value from enterprise information. ■

References

- Beck, Kent, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas [2001]. "The Manifesto for Agile Software Development," <http://agilemanifesto.org>.
- Hunt, Andrew, and David Thomas [1999]. *The Pragmatic Programmer*, Addison-Wesley Professional.
- Jeffries, Ron, Ann Anderson, and Chet Hendrickson [2000]. *Extreme Programming Installed*, Addison-Wesley Professional.
- Murphy, Craig [2004]. "Adaptive Project Management Using Scrum," *Methods & Tools*, Volume 10, Number 4 (Winter), pp. 10–22. <http://www.methodsandtools.com/PDF/mt200404.pdf>

(#17570) Reprinted with permission from Business Intelligence Journal, Volume 13, No. 1. Copyright 2008 1105 Media, Inc. For more information about reprints from Business Intelligence Journal, please contact PARS International Corp. at 212-221-9595.