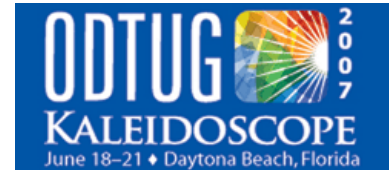


REAL TEMPLATES AND TIPS FOR DATA WAREHOUSING — THE AGILE BUSINESS INTELLIGENCE LIFECYCLE (ABIL)

Jeffrey Bertman, DataBase Intelligence Group (DBIG)



Overview

It's time to take a deep relaxing breath. Your data warehouse abstract is approved, or perhaps you're engaged to rescue a current project in distress. Either way, you can see the sun rising in the horizon. You daydream about the legends your team will make, and a grin bursts out uncontrollably on your face. Perhaps you've already implemented successful business intelligence projects, so you feel extremely confident in the upcoming results. Perhaps this is your first venture into the world of decision support, but your imagery skills are sharp as you envision a blue promotion ribbon waving on the door to your new window-view corner office. This will be the epitome of a well managed, cost-effective project generating much appreciated increases in your organization's revenue, profit, or market share. Or if this is your parallel universe government job, you may be increasing cost efficiency, detecting fraud and abuse, improving quality, or facilitating information sharing.

Euphoria is in abundant supply... until your eyes pass over the email from upper management. It's time to submit the project strategy, timeline, staffing plan, and budget. To settle on these matters, you need to decide upon the overall project framework. This will also help delineate how people within and external to the project interact with each other, and how interim and final work deliverables will be organized and implemented.

This paper helps restore your path to euphoria and confidence. It explains the most critical elements of successful business intelligence projects. It maps out a proven framework for producing full scale data warehouses and operational data stores with consistently reproducible — successful — results. Best practices and valuable tips and techniques are revealed to avoid common traps and pitfalls. Real world templates are also included to levy this paper as a complete toolkit.

The project framework and best practices presented here are collectively referred to as the Agile Business Intelligence Lifecycle (ABIL). It is worth noting here that in the world of operational systems development, agile methodologies have risen from ridicule to mainstream popularity, based largely on their widespread history of success when technical and cultural principles are properly blended. There are many agile methodologies to choose from, depending on the size of your team(s), the criticality of the applications you are designing and building, the culture of your business and IT organization, and other factors. Unlike the origins of operational systems development, business intelligence (BI) projects have tended toward agility since the formalization of data warehousing in the 1990's. Two main characteristics of most BI projects have been smaller team sizes and the breakdown of major deliverables into multiple mini deployment lifecycles or *iterations*, each of which *evolves* the system into progressively more valuable states. There are major distinctions, however, between iterative and agile thinking and doing. ABIL is a pioneer in filling the gap between iterative and agile for BI projects, and it provides a comprehensive model spanning the entire BI lifecycle across numerous increments in system value. In the course of explaining the key traits of ABIL (and other agile methodologies not focused on BI), this gap will be revealed. More importantly, you will understand the major credos and principles of agile development, and how to leverage them for repeated success in your IT environment.

What It's About

Preliminary focus of this paper covers the *what* picture -- *what* ABIL is about. Then we'll springboard into the details of *how* to leverage ABIL to achieve success in your BI projects. In the tradition of most agile methodologies, we'll present early on the major principles of the ABIL manifesto. And since the sharing of ideas is often best initiated with a picture, we'll present the lifecycle diagram.

Laying the Foundation — Major Terminology and Simple Best Practices

Before we dig into the fun stuff, let's make sure we are all on the same page regarding some basic terminology. It is quite possible for the same term to have somewhat varied and subjective meanings under different contexts, especially in the IT arena. So even if you feel comfortable with the agile world, spend some time reviewing the following terms.

- **Agile:**

This term is easy to define outside the realm of IT. Within IT and software development in particular, the word *agile* is typically defined in terms of its multiple principles. Providing a short description for this term can be misleading, but we'll venture a go at some synonyms:

*Lean and flexible, adaptive, physically and mentally acute and quick,
and well-coordinated through relatively natural means.*

Refer to ABIL Principles in the next section for a better understanding of how this definition applies to data warehousing / BI development projects.

- **Business Intelligence (BI) and Data Warehousing (DW):**

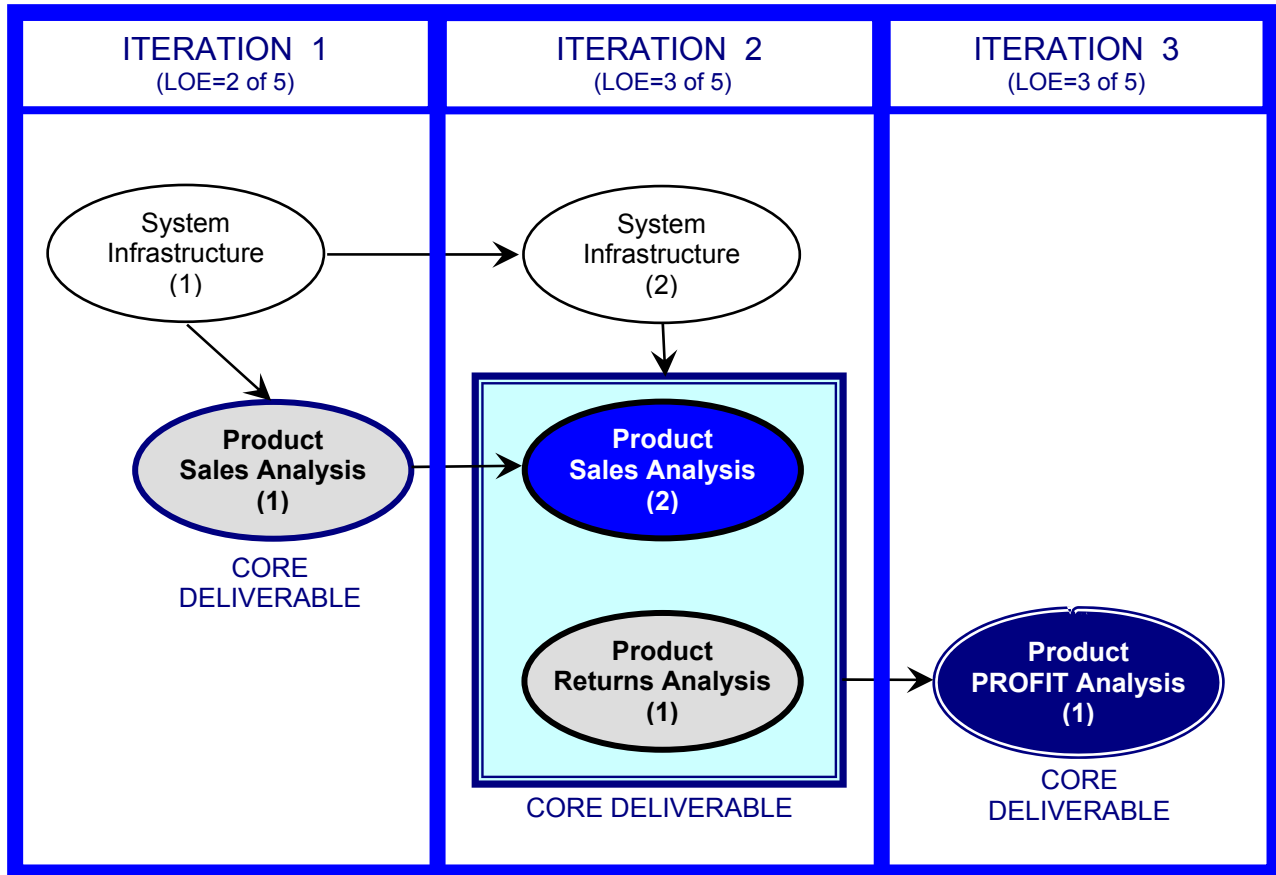
In the popular domain, these two terms are basically synonymous. For the remainder of this paper, we'll take the popular approach and use these terms interchangeably. For the sake of accuracy, a small distinction is offered here. Essentially, BI is the science of facilitating and improving strategy, decision support, and overall or targeted effectiveness of an organization. Data warehousing is currently the primary vehicle which enables BI. DW usually implies the physical consolidation of data to facilitate the use of analytical and mining tools to achieve BI. As time goes by, however, the concept of *virtual* data warehousing will grow, perhaps driving more of a wedge between BI and DW. A virtual DW makes multiple disparate data sources appear as though they have been unified, without actually consolidating and duplicating the data into a physically central database. Virtual warehousing is growing in popularity, enabled primarily by Service-Oriented Architecture (SOA) and other data integration technologies. However, to perform advanced analytics, physically separate data warehouses and repositories will remain sensible for at least the next five years. Moreover, as long as virtual data warehousing is implied within the context of overall data warehousing, DW should remain synonymous with BI.

- **Ceremony:**

The extent of detail and precision applied to the various interim work products required by a project lifecycle methodology. For example, lite ceremony during the logical design stage might involve writing use cases on a white board and saving them via digital photographs. Very simple and few standards would accommodate the creation and review of the diagrams. Heavy ceremony would involve making a high level context as well as detailed use cases in a formal modeling tool such as Rational Rose Modeler or Rational Software Architect. Furthermore, projects utilizing heavy ceremony often require more kinds of diagrams and system specs than in a lite ceremony environment. Ceremony also applies to project planning and coordination.

BEST PRACTICE TIP: Lite ceremony might involve a simple project map as shown in the figure below, depicting only the major component deliverables for each project iteration or release, and the basic dependencies between component sets across the iterations. You'll notice that the example figure also uses line style and shading to convey incremental levels of value, e.g. thicker and darker indicating more value. A project map may also indicate approximate level of effort (LOE) involved in each iteration, although this basic metric may not correlate directly to the number of delivered components. The complexity of each component is generally not indicated, and process maps generally do not show who does what or how long each lifecycle stage will take.

BEST PRACTICE TIP: A heavy ceremony example would be a more comprehensive project plan. While basic dependencies are sensibly tracked, most agile environments disdain fine grained PERT charts. As Craig Larman explains in *Agile & Iterative Development – A Manager’s Guide*, “A PERT chart is built on the assumption that the tasks of a project can be identified, ordered, and reliably estimated, that there is minimal change and noise in the system, and in general that a defined process can be applied. This is inconsistent with the recognition in iterative and agile methods that software is semi-chaotic new product development with high degrees of change and noise, and defined processes can’t apply.” You’ll see that the templates offered by ABIL allow tracking of dependencies in the most relevant places, e.g. project maps and plans, and action boards which clearly delineate major dependencies required to resolve an issue.



Example Project Map (use of different colors and line thickness are optional)

BEST PRACTICE TIP: In general, the degree of ceremony appropriate for a project is directly proportional to the combination of criticality and project size. Project size can be further broken into project scope, staff size, and the minimal complexity of each meaningful iteration. These sizing measures are potentially confusing, and often a major factor behind poor scheduling and team sizing. One premise of agility is that large problems can be broken into small incremental deliverables, each providing significant value to the users, and each substantial enough to validate success and reveal when future iterations require alteration. In the Example Project Map above, Product Sales Analysis can be broken into two iterations, and the value provided by each is deemed substantial. In iteration 2, however, either the second increment of Product Sales Analysis was either too small to fill a reasonably sized iteration schedule (maybe it takes just a few days to complete), or more likely it was too small by itself to qualify as a reasonable value increment. It was therefore combined with the first increment of Product Returns Analysis. When this manner of compounding becomes significantly large, it is usually appropriate to form multiple teams. In this way, even agile projects may grow to hundreds of team members spread across many separate but closely collaborating teams. For such large project environments, you may be tempted to revert to traditional lifecycle methodologies, but note there are several agile methodologies that can scale accordingly, e.g. Unified Process (UP), or to some extent, Scrum. BI projects naturally

tend to provide meaningful value with small iterations. Nonetheless, ABIL is able to scale to very large teams if there is a compelling reason to do so.

- **Criticality:**

The degree of negative impact or damage that would occur from system failure. The term *mission critical* generally refers to high criticality.

BEST PRACTICE TIP: Instead of assigning generic criticality categories such as low, medium, and high (mission critical), define more descriptive classifications. For example, renown agile author Alistair Cockburn offers the following: loss of comfort, loss of discretionary money, loss of irreplaceable money, and loss of life. Additional or tangent classifications are possible. In some government projects, for example, we might have a classification for loss of national security. This classification might be used in addition to loss of life, to help understand the full impact of failure.

- **Iteration, Sprint, Spiral, and Cycle:**

Across various agile software development methodologies, all these terms are basically synonymous with system increment or release. This refers to the completion of a full set of project tasks and work products, all the way through to a final functional deliverable which is generally deployed as a pilot or production system. With each increment, the system evolves and matures, providing progressively more value to the stakeholders and overall business community. The amount of value provided with each release is limited by the *timebox* (duration) of the increment.

BEST PRACTICE TIP: Keep iterations short, primarily to enable stakeholders and designers to verify planned versus actual functionality as early as possible, and to alter course when necessary for subsequent iterations. On the other hand, if an iteration is too short, the ratio of core work tasks versus checkpoint and review tasks is too low, ironically causing the ceremony and coordinative project infrastructure to consume commensurately excessive time. Different agile methodologies have different standards for the word “short”. For example, the following table shows the typical length of each system release or iteration for various agile methodologies.

Agile Methodology	Dev Focus (Operational or BI)	Typical Cycle Length	Term which Equates to Cycle or Iterative System Release
Evo	Operational	5 days	<i>Cycle</i> or Timeboxed <i>Iteration</i>
Extreme Programming (XP)	Operational	1 to 3 weeks	Timeboxed <i>Iteration</i> (or <i>Sprint</i>)
Scrum	Operational	30 days	<i>Sprint</i> or <i>Spiral</i>
Unified Process (UP) and Rational Unified Process (RUP) → Forms of Use Case Driven Dev (UCDD). → Also see OpenUP (and Eclipse Process Framework (RUP) and Agile Unified Process (AUP)	Operational	2 to 8 weeks	Timeboxed <i>Iteration</i> (or <i>Sprint</i>)

Agile Methodology	Dev Focus (Operational or BI)	Typical Cycle Length	Term which Equates to Cycle or Iterative System Release
Feature Driven Development (FDD)	Operational	1 to 2 weeks (sometimes monthly)	Timeboxed <i>Iteration</i> (or <i>Release Cycle</i>)
Agile Model Driven Development (AMDD) → Traditional MDD –But– “models which are just barely good enough” (www.agilemodeling.com/essays/amdd.htm).	Operational	5 days	Timeboxed <i>Iteration</i> Minutes of “Model Storming” + Hours of Test Driven Dev + <i>Optional</i> Reviews for QA and Reflection
Agile Data Method (ADM) → Initial attempt to focus AMDD on Data-Centric systems (www.agiledata.org/essays/vision.html).	Operational & Some BI –but– Data-Centric	1 to 3 weeks	Timeboxed <i>Iteration</i> Minutes of “Model Storming” + Hours of Test Driven Dev + <i>Optional</i> Reviews for QA and Reflection
Agile BI Lifecycle (ABIL)	BI (Data, Processes, and Services)	35 (to 65) days	Timeboxed <i>Iteration</i> (or <i>Sprint</i>) Cycle time depends on Meaningful Value Chunks (and Holidays)

Contrary to some extreme agilists, the first one or two iterations of a project might be longer than subsequent iterations, allowing for project personnel to ramp-up and for the framework to “gel”. This extra “initial warm-up” or ramp-up time is formally recognized in ABIL (e.g. for BI goal setting and analytical mapping) and in AMDD (for initial requirements and architecture modeling).

NOTE ABOUT VARIOUS AGILE METHODOLOGIES: Evo, XP, Scrum, UP, RUP (and variants), FDD, AMDD, ADM, and ABIL are different methodologies which all fall under what is commonly known as the Agile Umbrella. They all share core agile principles but practice them in different ways, sometimes for different reasons. ABIL, for example, is the only agile methodology tailored for analytical, business intelligence projects out of this list of popular variations in the operational arena. Many agile methodologies bring additional values and principles to the table, above and beyond the Agile Manifesto. Only ABIL is elaborated in detail in this paper, although the other methods are used occasionally as a frame of reference or to provide additional examples or insight into agile thinking.

- **Lifecycle:**
The roadmap of various stages through which an IT project passes. For the purposes of this paper, the scope is limited to BI design and development projects.
- **Methodology:**
A way to do something, regardless of how simple or complex. Agile methodologies are in the simple camp. In the words of Alistair Cockburn, renown author of *Agile Software Development—The Cooperative Game* (and other published works on agile development), methodology is “how [organizations] do business”.
- **Milestone:**
An event during a project lifecycle indicating when one or more specific tasks and/or work products are due or have been delivered.

BEST PRACTICE TIP: Even if the tracking method is as simple and infrequent as a post-iteration review (aka *post reflection*), planned versus actual milestone dates should be recorded and used to help improve future scheduling

estimates.

- **Reflection:**

A special kind of review practice. Reflection is the practice in which a group of people review their own processes and methods for accomplishing something, typically the achievement of stated goals. Reflection usually requires a level of objectivity, although agile methodologies recognize that subjective reflection is often just as valuable.

BEST PRACTICE TIP: Best practices regarding reflection are discussed later in this paper.

- **Role:**

Set of tasks and activities related to a single general purpose. Stakeholder (below) is one role, for example. Other roles include project manager or coordinator, team lead or coordinator, designer, developer, tester, deployer, maintainer, etc. In a small team agile environment, one person will often wear more than one hat (role). For example, the project manager/coordinator may also be a hands-on technical designer and developer.

BEST PRACTICE TIP: Role responsibilities in an agile environment may also differ from those of conventional projects. For example, a team lead/coordinator serves the team members, not vice versa. If he or she is failing to shield the team members from certain politics or fails to promote collaboration and productivity, the team members may openly request a vote to change the lead.

- **Stakeholder:**

Stakeholder is a person or business group that has an investment or other vested interest in the success of a project. Project sponsor is sometimes used synonymously with stakeholder. Stakeholders and sponsors may be further qualified, as in executive stakeholder, external sponsor, etc. A corporate or external sponsor is generally an entity outside your business organization, a special customer, for example.

BEST PRACTICE TIP: While stakeholders are generally not members of the technical staff, they should be considered as part of or at least an extension of the project team. Stakeholders should be informed and collaborated with regarding issues which may significantly affect system quality, reliability, serviceability, or release schedules. They should not be burdened with most technical issues. For this reason, it is important to separate *management* from *technical* issues in periodic status meetings, action boards, and such.

- **Standard:**

A convention, rule, or covenant used to guide the way a task or deliverable is performed, to promote a predictable result or model of comparison. In a means to end model (which applies to most software lifecycles), standards can address both means and ends. Most standards are documented or at least sufficiently descriptive to convey the desired information across multiple parties, so anyone adhering to the standard can easily achieve the same result.

- **Task (including different kinds of tasks, and tasks which are In and Out of Scope):**

Tasks are specific activities, duties, or units of work necessary to perform during the course of a project, for the fulfillment of some part of the project. To help distinguish a task from a completed deliverable, think of a design activity (task) which ultimately leads to a design specification (work product or deliverable). Core tasks refer to direct time spent on a work product or deliverable. Design, development, testing, and deployment are examples of typical core tasks. Coordinative or support tasks focus on supplemental (but important) activities such as project management, coordination, coaching, quality reviews and other checkpoints, etc.

BEST PRACTICE TIP: Everyone in a project team should constantly watch for activities that do not significantly contribute to the specific work products at hand. Such tasks are probably out of scope — or at least *outplan*, that is, outside the scope of the current project sprint or iteration. Outplan tasks from future iterations often try to creep into earlier project phases in the interest of providing certain value sooner versus later. Phase creep, if not managed carefully, encumbers the project schedule and quite often lessens the quality and maintainability of the final deliverables. Sometimes compelling reasons such as new government regulations or competitive behavior justify the need for outplan tasks. When this is the case, all team members and stakeholders should be notified proactively regarding the schedule

impact. This will help minimize the negative impact of rushed carelessness on quality and maintainability.

- **Team:**

A group of people working together to achieve common goals and deliverables. A general credo of most agile methodologies is minimalization — smaller teams of tightly collaborating people and simple and adaptive framework are more effective than larger teams with strict, definitive, and potentially cumbersome processes. Consequently some agile methodologies allow for only one or two small teams, with no more than 8 to 20 total people. Larger teams require more ceremony than these agile frameworks are designed to handle. By allowing the addition of multiple teams, some agile methodologies scale much larger. Scrum is one case in point. Promoting consistency and adequate information sharing across numerous teams, however, eventually requires the addition of more ceremony than that offered by many agile frameworks.

BEST PRACTICE TIP: There are some agile methodologies which can apply the principle of adaptation to more than just project coordination — they can adapt to projects with extremely high criticality and/or numerous teams, essentially by scaling the degree of ceremony. One such case is the Unified Process (UP), which “can be applied to small three-person projects with [low criticality], and [can scale] up to hundreds of developers working on life-critical systems.” (according to Craig Larman in *Agile & Iterative Development – A Manager’s Guide*). UP accomplishes this scalability by drawing as needed on a repository of processes and templates collectively referred to as the Rational Unified Process (RUP). ABIL is another such adaptive framework with an optional set of templates to draw on as required to handle progressively larger and more critical project environments.

- **Work Products and Deliverables:**

Work products are materials we publish or deploy in some way as either interim or final deliverables during various stages of a project lifecycle. Work product and deliverable are basically synonymous. A system design is an interim work product, which is also commonly referred to as an artifact. It may be simple (e.g. handwritten on paper napkins), or more formal or complex (e.g. output from a design or modeling tool). Completed, tested source code could be considered a final work product, although contractually an IT project might be obligated to deploy a system to production before it is considered complete.

The ABIL Manifesto

The agile world is already replete with manifestos and credos, most notably the Agile Manifesto at www.agilemanifesto.org and its related Agile Principles at www.agilemanifesto.org/principles.html, as outlined below.

The Agile Manifesto			
#	More Value	over	Less Value
1)	Individuals and Interactions	over	Processes and Tools
2)	Working Software	over	Comprehensive Documentation
3)	Customer Collaboration	over	Contract Negotiation
4)	Responding to Change	over	Following a Plan

The Agile Manifesto (www.agilemanifesto.org)

The Agile Principles	
1)	Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2)	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3)	Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4)	Business people and developers must work together daily throughout the project.
5)	Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6)	The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7)	Working software is the primary measure of progress.
8)	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9)	Continuous attention to technical excellence and good design enhances agility.
10)	Simplicity — the art of maximizing the amount of work not done — is essential.
11)	The best architectures, requirements, and designs emerge from self-organizing teams.
12)	At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

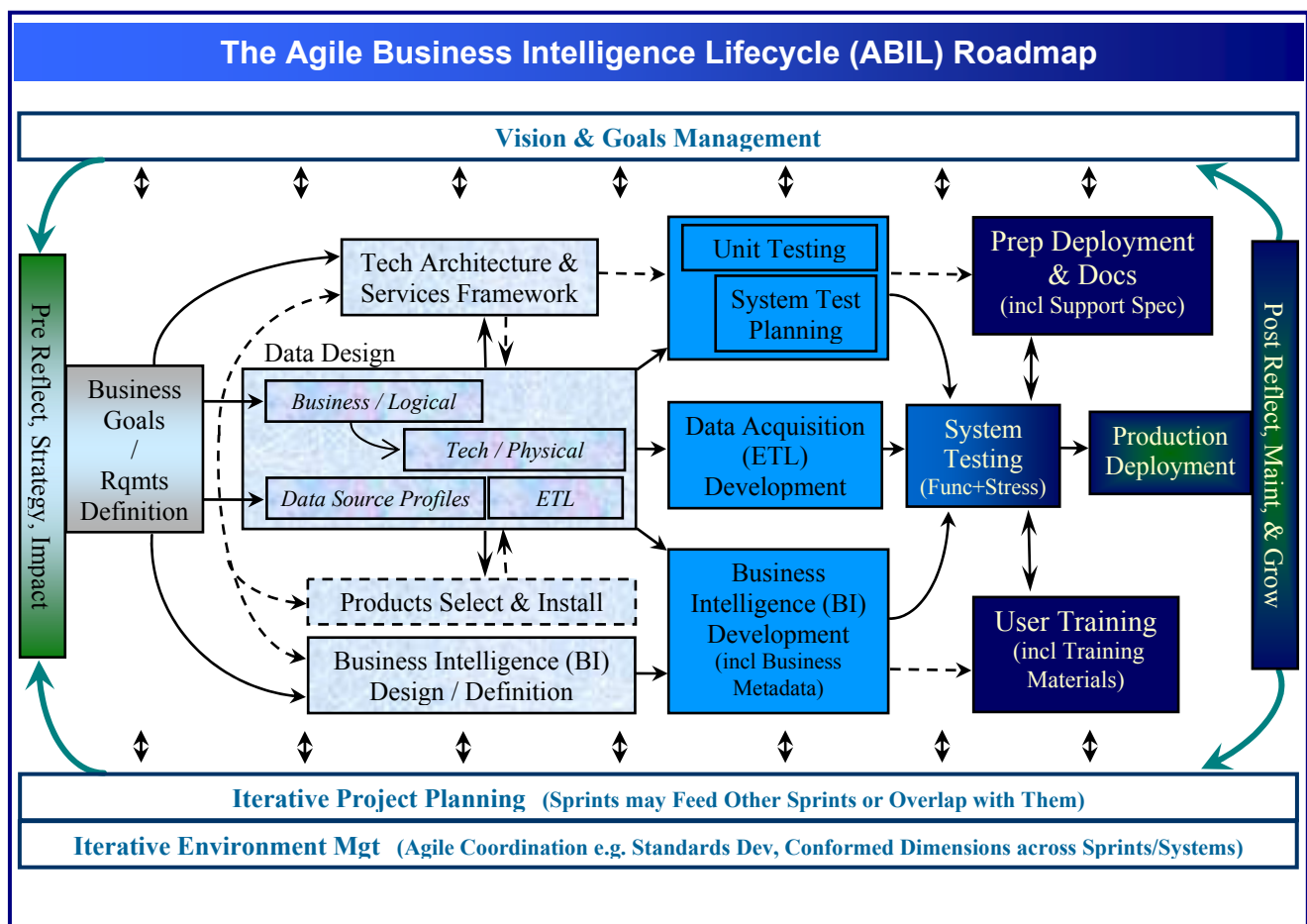
The Agile Principles (www.agilemanifesto.org/principles.html)

The ABIL Manifesto is essentially layered on top of the basic Agile credos. Since we still have a lot of ground to cover in a relatively short space, this manifesto has been leaned down to allow sufficient space to present various work products and related ceremony. Many agile proponents believe there is contradiction in an agile manifesto accompanied by structured templates and such. Remember that agile environments offer sensible alternatives to thick ceremony, most notably adaptability and close and frequent collaboration. So ceremony should be added only as needed, e.g. to accommodate large projects as explained in the previous section. Also keep in mind that the work products presented here for the most part serve as examples and simple checklists. It is up to each project team to decide what level of structure and depth to apply at the ceremony level. It is entirely possible for different teams within the same project to exercise varying degrees of ceremony, although certain standards of collaboration and information sharing should be implemented.

Agile Business Intelligence Lifecycle (ABIL) Manifesto			
#	More Value	over	Less Value
1)	Start with the Agile Manifesto	over	Thick Prescribed Management and Methodology
2)	Pictures	over	Words
3)	Just Enough (simplicity)	over	Comprehensive Project “Shelfware”
4)	Just in Time Task Parallelization (overlap)	over	Stretched out Serial Lifecycle
5)	Short Manageable Increments of Meaningful Value	over	Large Releases and Artificial or Political Deadlines
6)	Progressive, Maintainable Value	over	Front-heavy Value

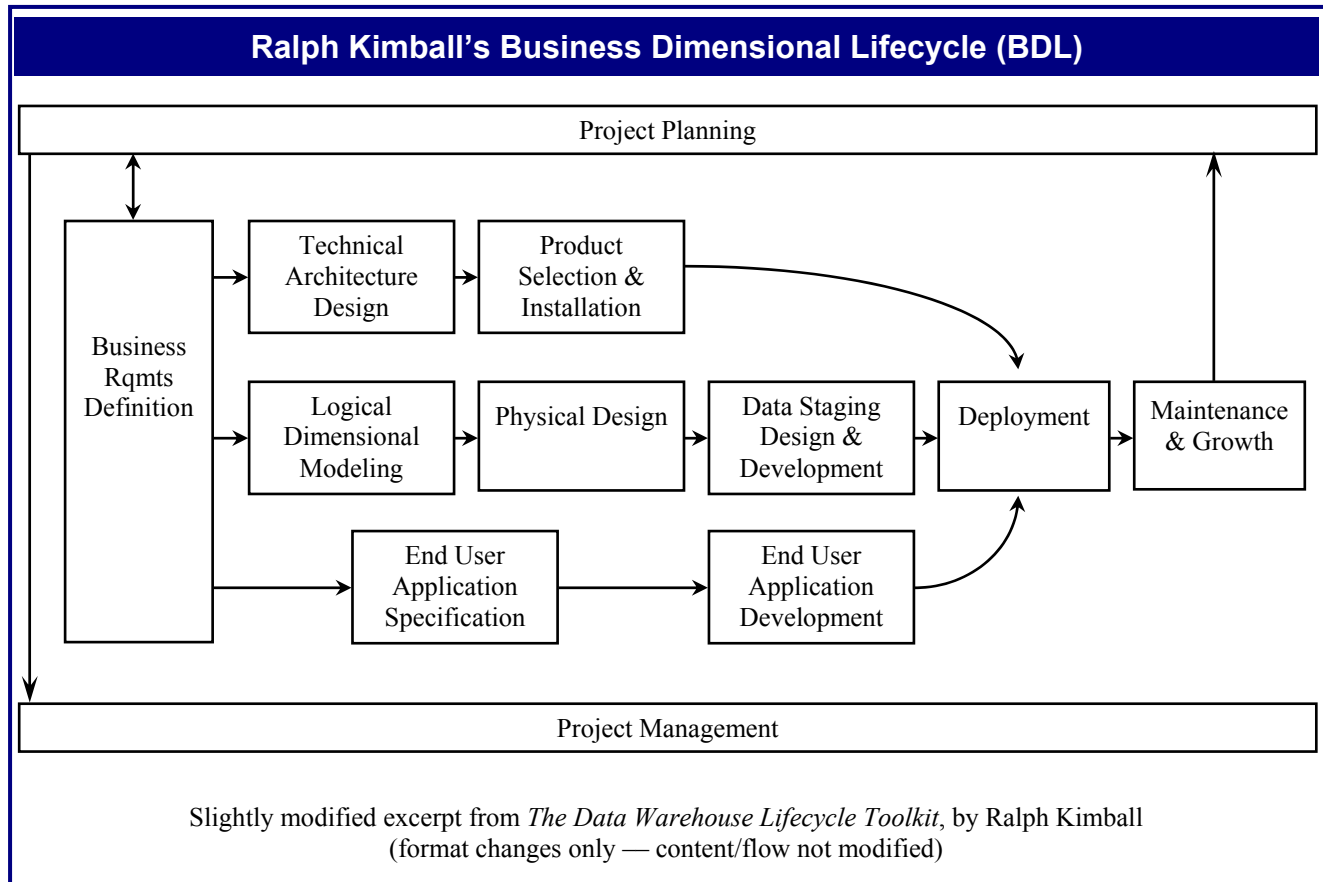
Agile Business Intelligence Lifecycle (ABIL) Manifesto			
#	More Value	over	Less Value
7)	Adaptive Course	over	Stringent Strategy
8)	Adaptive, Scalable, Balanced Methodology	over	Extreme Agility
9)	Continuous Improvement using Simple Tracking and Subjective Means	over	Detailed Productivity Tracking

The ABIL Manifesto



The ABIL Roadmap

The Agile Business Intelligence Lifecycle has been proven in the design and development of numerous successful data warehouse projects. It's roots were formed in Ralph Kimball's Business Dimensional Lifecycle (BDL), as shown in the figure below.



Ralph Kimball's Business Dimensional Lifecycle (BDL)

How ABIL Works

Due primarily to space limitations, this paper highlights certain notable aspects of ABIL in lieu of comprehensive ramification. Ironically, this makes the paper itself lean in the agile direction :-).

Roadmap Highlights

- Color and Texture:**
 Early lifecycle stages are portrayed in light colors which transition into darker, more pronounced colors as value is added during the lifecycle to achieve the various interim and final deliverables. For example, notice that goal setting and requirements definition are shaded in light grays and blues, while training and deployment related tasks are in dark blue. The architecture and design stage is depicted with a medium blue texture, implying that this is where most of the definition work occurs.
- Reflection on Both Sides and Throughout the Lifecycle:**
 It is traditional to see a post implementation review stage at the end of an system release or iteration, but how often have you seen this practice overlooked?

BEST PRACTICE TIP: Whether due to the elation of completing a significant release or the pressing schedule of the next release, post implementation reviews are often skipped or poorly documented. As indicated by the numerous little

two-way arrows across the top and bottom of the ABIL Roadmap diagram (beneath Vision and Goals and above Project Planning), one common agile practice is to have all team members periodically reflect on perceived productivity and effectiveness. Scrum projects, for example, typically review these items during biweekly informal stand-up meetings. ABIL purports the same practice, and adds a little more objective. These sessions should also sanity check ongoing alignment with the organizational and project level vision and mission statements. Moreover, in order for lessons learned to persist as team members come and go on the project (and certainly across multiple projects in your enterprise), reflections should be more formally documented, at least at the end of each iteration. In ABIL, we acknowledge that traditional post-release reflection does not always work. Consequently we remind ourselves at the start of each iteration to document the lessons. Since we are agile, remember that the documentation should be light. For example, you may add a few items to the management action board, outlining simple improvements to existing methods of collaboration or other parts of the lifecycle.

- **Task Parallelization (Overlap):**

The concept of multi-tasking is certainly not new, but how this concept is applied in ABIL versus traditional serial development environments is markedly significant. Even in Ralph Kimball's Business Dimensional Lifecycle which is clearly iterative, notice how Logical Modeling, Physical Design, and Data Staging Design are all strictly serialized (aka waterfall) after one another. A methodology with relatively short iterations is not necessarily agile compliant.

BEST PRACTICE TIP: Now look at the corresponding elements of the ABIL Roadmap and notice how they overlap. The overlap is not mandatory, but it is clearly evident and even fostered. The primary benefit of partially overlapping these tasks is that we eat our own dog food sooner (to coin a Microsoft phrase), and therefore discover problems sooner versus later. The downside is that some problems involve re-writing some of the work developed in subsequent phases. With a technically aware and communicative project or team coordinator, the tradeoff is usually beneficial. Remember, in agile environments management personnel focus more on team and activity introspection and interaction than on forcing rigid task boundaries. This motif promotes effective impact analysis and balances the outcome in our favor.

Ceremony Highlights & Templates

PLEASE EXCUSE THE DUST...

This paper is version 1.0 and has been abbreviated to fit into the Kaleidoscope 2007 Proceedings Manual. See upcoming releases at www.odtug.com (look for Kaleidoscope 2007+) and the ABIL Manifesto website (www.ABILmanifesto.org coming soon). Or contact the author directly per contact information provided in *About the Author* section at the end of the paper.

Also see the Kaleidoscope 2007 PowerPoint Presentation for this paper. It includes more templates, and also reviews special data warehousing topics such as building data marts before or after the warehouse, real-time versus periodic ETL, workflow scenarios, push/pull frameworks, e-business considerations, data quality management, and platform independence.

About the Author

Jeffrey Bertman is CTO and a principal consultant for DataBase Intelligence Group (DBIG), supporting customers at both strategic and operational levels for the enterprise and targeted systems. Specializing in mission-critical services and decision support, Mr. Bertman has over 18 years of hands-on experience and has been designing and building comprehensive database environments and application systems since 1987. He has published articles and developed courseware on a wide range of technical and management topics including data warehousing and business intelligence, database administration (DBA), high availability, disaster recovery, business process management/engineering (BPM), IT organization, lifecycle optimization, and project management. Business disciplines include accounting, telecom, health care, marketing, and manufacturing applications. Mr. Bertman can be reached via e-mail at JeffLit@dbigusa.com.