Business-Model-Driven Data Warehousing
Keeping Data Warehouses Connected to Your Business

Dr. Hakan Sarbanoglu
Chief Solutions Architect
Kalido

Bruce Ottmann
Principal Solutions Architect
Kalido
Table of Contents

Introduction 4

Conventional DW Design Process 5
  Disconnect #1: Business Requirements 5
  Disconnect #2: Agreement on Logical Model 5
  Disconnect #3: Understanding the Physical design 6
  Disconnect #4: DBAs perform fine-tune editing on the design 6
  Disconnect #5: Changes to the design during the development 7
  Disconnect #6: The data modeling tool is offline 7
  Disconnect #7: Semantics were left in the business dictionary 8
  Disconnect #8: Inadequate master data management practices 8
  Disconnect #9: Inconsistencies in the BI layer 8
  Disconnect #10: Responsiveness to business change 9

Business-Model-Driven Data Warehousing with Kalido 10
  What is a business model? 10
  Data warehouse development chronicle with business modeling 12
  Avoiding the Business-IT gap with Kalido data warehousing 15
  Benefits of Business-Model Driven Data Warehousing with Kalido 18
Introduction

The quality of Business Intelligence (BI) depends mostly on the available information behind it. This is particularly evident given such functional improvements in the latest generation of BI tools as much better reporting and analysis functions, new visualization capabilities, dashboards, portals, data mining tools and new analytical applications like business performance management, forecasting, planning etc. Unfortunately, business users are still dissatisfied by the business information they receive every day. What the business needs is consistent, accurate, accessible, actionable and transparent information across the whole company.

Today’s businesses also need flexibility and responsiveness, so that when they want to extend or change their BI – either because they have a different view now or because of an actual business change – this should be quickly achieved without any compromises on the quality of BI.

Why is this so difficult? There is no doubt about the importance and value of BI, and we have made an enormous progress in this field. At the same time, we are still struggling with issues like inconsistency between two reports that we received in the same morning, arguing over the completeness of data, or dissatisfied that this was exactly what we needed. There are times when we want to trace back what changes have caused our figures to appear so different this month or to audit whether our business rules are consistently applied.

Data quality is the usual suspect, but equally, we should also question the effectiveness of our data integration in data warehouses and marts. It is difficult to achieve total integrity when the business rules used for derived data are spread across multiple ETL strings as on-the-fly transformations or applied as last minute transformations in the fragmented BI components.

IT has rightly turned to enterprise data warehouses and master data management (MDM) to fix these issues. Regardless of whether they are sitting on a highly scaleable platform, implemented as a central or federated architecture, or designed in normalized or dimensional models, most of enterprise data warehouses – and the data marts derived from them – fail to deliver fully on their promises to the business.

This paper analyzes the issues of conventional data warehouse design process and explains how this practice can be improved using a business-model-driven process that covers not only the requirement definition and design, but also the development, operation, maintenance and change requirements of the data warehouses and data marts in support of effective BI.
Conventional Data Warehouse Design Process: Lost in Translation

Each data warehousing (DW) project starts with a joint IT and Business effort to define the business requirements. In the traditional approach, however, IT and business struggle to obtain good requirements because they do not have tools that can facilitate communication in a way that both sides can follow and understand. This disconnect continues all the way through the implementation process and into the operation of the system.

Let’s look at the areas of disconnect in this process.

Disconnect #1: Business Requirements
Business representatives in DW and BI projects usually document their requirements with free texts, lists, unstructured diagrams, report descriptions and sometimes with conceptual diagrams. This phase is driven by the business users themselves or BI directors responsible for delivering reporting and analysis capabilities to end users.

These people are usually comfortable with defining their requirements in business terms in the project definition. However, even when using the business language with which they are familiar, there can still be significant disagreement and conflict on the business side. Typical issues include different views between departments such as different classifications of the same things, unclear (or lack of) business definitions and inconsistent or missing business rules. Nobody should expect 100 percent accurate business requirements in the first round. Experience has proven that attempting to finalize the requirement definitions as a distinct phase and handing over a frozen version of requirements to IT (as done in most ERP projects) is unrealistic for DW/BI projects.

Data Architects and DW/BI designers may start getting involved in this phase, but they generally remain in the background, trying simply to understand the requirements as much as they can. Business requirements remain loose, and despite all the attempts to the contrary, cannot be tied down in a single meeting. Most such issues are identified and logged for resolution in later phases.

Disconnect #2: Agreement on the Logical Model
The data architect combines the initial understanding of business requirement (however incomplete) with his/her analysis and profiling of the available source data and creates a logical data model. Typically, a gap is identified between the defined requirements and the available data in the source systems. The data architect and business representatives come up with some compromises on the requirements and the data model. This data
model is documented mostly using a graphical data modeler tool (such as Erwin, ER Studio or PowerDesigner) which is then verified with the business representatives.

Here, we see clear signs of a problem. It is a “data model,” and the business needs some explanation about what they see in these technical diagrams. Already unsure about the exact details of their business requirements, the business representatives tend to make just a few comments and corrections, then, without a full understanding of what they have defined together, the data architect and the business representative, “agree” on the logical model so that the project can progress.

**Disconnect #3: Understanding the Physical Design**

The next step is to convert the logical model to a physical database schema design. Data modeler tools can convert it to an initial physical design and then database designers engage in the art of physical design by adding/changing indices, foreign keys, declaring the primary keys and data types, or de-normalizing/normalizing tables as needed.

At this stage, the architect may apply well known data warehouse design best practices (generally following patterns advised by Bill Inmon or Ralph Kimball). This means that the data-warehouse-oriented design becomes somewhat different from the logical model communicated to and agreed with the business representatives. Most of those design techniques are so complex that there are still long discussions about them between the data warehousing professionals. And, nobody can expect the business representatives to understand technical concepts like “junk dimensions” or “bridge tables” that have no meaning to the business inside a BI report or analysis. There may be a few rounds of further interaction on these designs with the business, but now facing even more complex IT details, business representatives already feel that the rest of the development process should be in IT hands. The first version of the design is sealed in the modeling tool, which generates an impressive database creation script. Although agreed with a handshake, both Business and IT knows that they are extending their hands over a gap. This milestone in the project also represents the point where the data architect hands over the responsibility of the design to a separate development team.

**Disconnect #4: DBAs perform fine-tune editing on the design**

The design is then taken over by the development team to build the data warehouse. First, the database creation script should be run against the DBMS to create the data warehouse physical schema. At this stage, database administrators (DBAs) make additional edits (before or after running the database creation script) for standardization or performance fine tuning purposes. Edits may also be required to adjust the schema for the underlying
DBMS and hardware platform. The Business has no visibility into these changes because the changes are deemed to be just “technical” in nature. However, those changes may unwittingly have an impact on the business requirements.

**Disconnect #5: Changes to the design during the development**

During the development phase, we may still need to change the design many times for technical reasons, as well as for changes to business requirements or to reflect a better business view. Both may cause further disconnection.

Once the initial loading starts, and the tables are filled with actual data, we discover issues like cardinalities in the data are not as we intended, or duplicate identifiers, or non-standard data values. This could be minimized with profiling before the design. This typically results in minor changes like adding a flag column or a new identifier.

Nevertheless, on a regular basis, additional changes are requested by the business team who – hopefully – are still actively involved during this phase. Most people agree that the development should follow an iterative method, so early prototypes or identified issues are resolved with the business representatives. In so doing, they may propose not only minor changes, but sometimes new attributes, new source data or even new hierarchies. The big question is where all those changes should be applied during the development. Should the project manager have them incorporated into the data modeling tool and then re-create the development instance? This is first a matter of change management discipline. Very few go back to the data modeler, often due to time or resource restrictions. Nearly all such changes identified during the development phase are directly applied on the development instance. When this occurs, the logical model becomes totally disconnected from the physical database, and we no longer have a logical record that ties to a physical instance.

**Disconnect #6: The data modeling tool is off-line!**

In today’s “data” modeling tools, making changes to the logical or physical model and then creating a “delta DDL” (data base modification script in Data Definition Language) is possible. But, there are changes that are not supported. Even if a delta DDL is created from these tools, it is unlikely that it would work on the development DBMS because of changes made directly to the actual instance. Sometimes, we even see an organizational disconnect between the data modeling tool and the data warehouse database. Graphical data modeling tools are for use by data architects in the Enterprise Architecture group whereas the development project team may not be using or given access to these tools.

The reality is that most of the data architects are not involved in changes subsequent to their initial design. The best organizations would still be keen to at least capture the as-built design back into the data modeling tool – perhaps by using reverse engineering – but such “best practice” still serves only documentation purposes. The data architect is disconnected from the as-built data warehouse. The business representatives have already been cut off because the IT folks are talking in a different language now; and they are waiting for the User Acceptance Test (UAT) anyway.

Changes proposed after the UAT, and then any changes implemented during the production life of a data warehouse, face the same treatment; they are applied in the DBMS platform, not in the data modeler tool.
**Disconnect #7: Semantics were left in the business dictionary**
Experienced architects and business analyst representatives are keen to capture the meanings of their business information objects and business rules. As a result, they usually document these in either the data modeling tools or, commonly, in a separate business dictionary document. Very few data warehouse projects convey these semantic definitions into their development instances.

Some are using meta data management tools, but mostly for technical meta data management in support of ETL mappings. Rarely do projects include automatic carryover of all these semantic definitions all the way from the modeling tool into the reports generated from the BI layer. This is because the end-to-end software platform from sources to BI is so fragmented in conventional data warehousing. The Business users see a complex and confusing set of terms in each aspect of the system and give up insisting on keeping the definitions and rule descriptions together with data.

In addition, the business rules specific to how things are done in a particular place, are often fixed in the structure of a data model. Rules are translated by database designers to data models. This means that small changes in the way business is conducted can lead to large changes in data warehouses. Even worse is where these business rules are embedded in transformation mappings in the ETL code and/or applied in fragmented BI components as a last minute transformation. All these practices are questionable from consistency and regulatory transparency perspectives.

**Disconnect #8: Inadequate master data management practices**
When data is received from multiple sources, we inevitably hit data quality and integration issues, particularly concerning the shared master data. Examples include incomplete, inconsistent and duplicate information about entities like suppliers, customers, products, KPIs, financial data or assets. Most data warehousing projects approach this issue from just a data quality angle and typically attempt to correct master data inconsistencies using ETL, unmanaged lookup tables and transformation rules. This is a major consistency issue for the business users expecting to receive a harmonized and complete view of their business when they receive their reports. They often find themselves in a situation where this inconsistency has been patched by ETL, so although their reports may seem consistent, they are not accurate and don’t reflect the business. Unfortunately, the users are not usually aware of this. Even if they know about master data management problems, the master data stewards do not have proper solutions to govern the master data about their business entities.

**Disconnect #9: Inconsistencies in the BI Layer**
Most BI tools are now offering a conceptual layer enabling the definition of business objects in business terms and the mapping of these objects back to the data objects in the warehouses, data marts or other sources using technical meta data and SQL. BusinessObjects Universes and Cognos Framework Models are good examples of these capabilities. These architectural components should be seen as a buffer between all the underlying complexity and the reports that users work with.

But this architectural layer is fragmented by its nature because it does not hold persistent data across the business. There are generally multiple universes, each designed for a limited sub-scope and certain group of users. There is no guarantee that a business rule hardwired in the SQL of an object is consistent with another SQL supposedly mapping to the same object in another report. Some BI tools provide the flexibility (or risk) of
having a separate conceptual layer component for each report. In general, these conceptual layers are defined with yet another interface where the semantic layer is re-generated from the technical meta data of the data sources. This simply adds to the complexity of the entire BI infrastructure. Furthermore, these conceptual layer components are independently maintained, so they may become disconnected from their main source warehouses and marts.

**Disconnect #10: Responsiveness to business changes**

The actual business world changes frequently. The business comes back to IT with new BI requests, such as a new analysis view of their sales and profitability, an organizational change, a new chart of account structure, or something even more significant, such as mergers and acquisitions. As a result, numerous parts of the data warehouse infrastructure need to be changed, and the business requires a rapid response. As importantly, the business wants to keep a corporate memory, so that they can meaningfully compare what they planned last year to the latest view of the actual business today.

How fast can IT respond to these change and extension requests? Weeks? Months? Can we keep our design and development principles during those rush periods? This is very difficult because the data modeling tools are already disconnected. If we are using ETL tools to do the heavy lifting of the data warehouse build and maintenance, the ETL tools, the underlying DBMS, the meta data management tool and the BI tools are all disconnected.

As seen in Figure-1 there are many user interface points with different tools used by separate role players with multiple disconnection points in initial and follow-up design of conventional data warehouses. It should not be a surprise to see why there is a gap between the business and IT in the conventional BI landscape.

As seen in Figure 1, there are many user interface points with different tools used by separate role players with multiple disconnection points in initial and follow-up design of conventional data warehouses. It should not be a surprise to see why there is a gap between the business and IT in the conventional BI landscape.
Business-Model-Driven Data Warehousing with Kalido

The Kalido Information Engine relies on the following key enablers:

- Business-model-driven software products covering end-to-end architecture between the data extracted from the source systems and the BI layer. The business model governs the whole life cycle including requirement definitions, design, build, deployment, operation and maintenance.

- Integrated and automated management of multiple data stores including staging area, generic storage area, data warehouse reporting schema and data marts. The generic storage area is particularly key because it enables the preservation of the historic context, simulating the future and providing enterprise-wide identification of objects. More importantly, it enables all changes to the model and data to be maintained using associative modeling and the principles based on ISO-18876 and ISO-15926 standards. This generic storage is used for storing metadata (including the business model itself) and the reference data by all software components.

- A business-model-driven capability to ensure the BI metadata and configuration for tools such as Business Objects and Cognos are kept in sync with the Kalido warehouse. Changes made in the Kalido business model automatically flow through to configure BusinessObjects Universes and Cognos Framework Models so that the time and effort associated with making a change in the warehouse is dramatically reduced, and no additional BI configuration, ETL or hand-coding is required.

- Master data management capabilities enabling the harmonization of common data about business entities using workflow-driven governance processes, so that dimensional reference data fed into the warehouse provides a consistent, complete and authorized view of the business context.

- The Kalido Approach as a best-practice services implementation framework for achieving rapid value from the Kalido Information Engine. This proven method facilitates thinking big, starting small, iteratively developing, following best practices like Agile methodology, deploying in a controlled fashion and then incrementally extending your scope, evolving toward an enterprise data warehouse.

Here in this paper, we will focus on managing the data warehouse life cycle process throughout the end-to-end architecture and discuss how this method can eliminate the disconnections.

What is a Business Model?

There are different definitions of the term “Business Model” depending on the context. In a business management context, the business model is understood as “a description of the scope and purpose and the way of doing business.” Hence, we talk, for instance, about the eBay, UPS or Google business models today. In a Business Process Re-engineering context, the business model means “a view of the activities and processes carried out by a business,” where they talk about how the business processes are executed and exceptions are handled by using graphical Activity Models together with Business Process Modeling techniques.

However, in the “information management” context, a business model is a view of the objects of interest to the business. In information management, the Business Model is simply the description of your business, its activities (business transactions like maintenance tasks or procurement orders), the measures about these activities that are used to monitor and manage them (such as gross sales, maintenance labor hours, volume and price of the ordered material etc.), the business contexts in which these activities take place (like facilities,
equipments, wells, customers, products, workers and suppliers) and finally, the business rules binding these things together.

Following the generic modeling principles, a business activity type (e.g. sales), is perceived as a separate entity type. Each occurrence of a business activity is a business transaction. A business transaction – for example, a particular sales transaction – is an event that happened at a point in time. In each event, we need to measure some quantities and qualities, like sales volume, agreed discount etc. In each event, a number of business entities are “involved,” for example, who the customer was, what product was sold, where the product will be delivered to, what the date of sale was. These are the business entities involved in a business transaction. While we carry out the business activities with pre-defined business processes, including exception handling rules, we capture certain facts of each business transaction (measure values and the identifiers of involved business entities) with a transaction record.

Figure 2: A sample Kalido Business Model
Those business entities involved in transactions are members of some classes (e.g. Person or Product), based on their defining properties. We tend to classify the members of a class based on some clear rules (e.g. credit rating of a person), so such groupings and taxonomies are also perceived as classes or hierarchies of classes. Associations between classes are not limited to just classifications. Entities may be linked to other entities of different classes with association types including ownership, installation, location, assignment, composition, etc. In most cases, such associations are born or changed as a result of a business event, like a sales event creating an ownership association between a product and a person.

**Data Warehouse Development Chronicle with Business Modeling**

A Kalido data warehouse project typically starts with a “business modeling workshop,” where the Data Warehouse Architect (now better called Information Architect and Kalido Modeler) leads a discovery session to identify the business activities in scope, to capture the required key performance indicators (measures), the business context and the business rules binding these entities.

Using the gesture-based graphical user interface of the Kalido Business Information Modeler software, the architect can draw the model with business terms during the discussions. This first modeling session may only define the main classes, transactions and key attributes and identifiers. At this early point, the business representatives will be able to see their business transactions, their business management hierarchies (like product classification or their internal organization), and visually following the associations. They will understand how they can dimensionally aggregate their raw data up to business analysis levels where they can compare to other data sets, like their targets – all in business terms. This first cut business model can be visually validated by the business in the first workshop.

The architect can assure the business representatives that they will follow an iterative development approach and that there will be continuous interaction to improve the model up to their satisfaction. Further details, like association types and cardinalities, alternative identifiers, attribute data types, Kalido Dynamic Information Warehouse (DIW) class of business entity (CBE) types, etc., can be added in the properties panels by the Architect outside this workshop.

The next step for the Architect is a gap analysis by working with the source systems specialists to see if the required data is available in source systems with appropriate quality and timeliness and whether derivations for key performance indicators are possible. Then the business model will be quickly improved with such resolutions and validated jointly by the architect and the business representatives. All model definitions are technically validated in the Kalido Modeler in real time to avoid surprises during its deployment.
When the model is ready for the first development iteration, the Kalido software automatically deploys it to a Kalido DIW instance. Kalido DIW imports this model, stores it as time variant metadata and automatically generates the warehouse reporting schema and data marts as a source to BI. As we’ll recall, the conventional data warehouse modeling method requires us to convert this business (aka conceptual) model to a logical model, then to a physical design, then generate a database creation script and then edit that script before creating the actual physical data warehouse schema. In Kalido, these steps are unnecessary: Once deployed, the warehouse is ready to load data. The business model also carries over all semantic descriptions to the warehouse.

Once the model is imported into the warehouse, it is stored in a highly stable, automatically generated, time variant generic data model which also stores the reference data.

When the model is imported, Kalido automatically creates a managed staging area to receive the raw data and a best practice reporting schema with the optimum level of dimensional de-normalization. The business model definitions dynamically drive all functions, including loading, validation, data derivation, summarization and data mart generation.

ETL is used only to bring the data into the staging area and not for transformations between the internal components from staging to BI. As a result, project teams can eliminate the redundant and costly ETL seen within conventional data warehousing. This removes the risk of inconsistency exposed by those ETL processes and provides transparency throughout the warehouse’s maintenance life cycle.

Having imported the model, we can immediately start loading reference data from a Kalido managed staging area, followed by transaction loading. At this point, the Kalido Architect or the Implementation Specialist will add some warehouse-specific object definitions, such as calculated measure definitions, transaction file definitions, summary transaction data sets for aggregate awareness, etc. We are then ready for to generate data mart, which are also defined as metadata in the warehouse user interface. The data marts are also defined and incrementally maintained using the business model.

Most data warehouses face the challenge of conforming the reference data, particularly when there are multiple sources. If the data about the same entities – products, for example – needs integration before loading into the warehouse, we can use Kalido Master Data Management (MDM). In this case, the business model that we deployed into the warehouse is pulled into Kalido MDM. Additional validation rules can be defined within Kalido MDM, such as data masks and auto-generated codes. The reference data from multiple sources are first loaded into Kalido MDM where they are validated on
the fly against the business model and then mapped and merged with other reference data using MDM functions. Invalid records are stored in MDM and enriched and harmonized through workflow-driven processes before they are finally authorized. Kalido MDM can be modeled to manage any subject area as well as used for authoring master data that does not exist in any source systems. These could be global key accounts or business classifications and taxonomies. Finally, authorized master data will be interfaced to the warehouse. Again, this whole process is incremental so that any modifications and additions will be processed through Kalido MDM for those classes under Kalido MDM’s control. In this way, Kalido MDM ensures the consistency, completeness and accuracy of business entities for BI analysis.

Once the initial population of warehouse is completed, maybe for a subset scope for the first iteration, we are ready to create first reports. These reports can be produced as Excel pivot tables generated from Kalido DIW. In most cases, the whole first iteration, including the requirement definition and modeling, should not take longer than 2-3 weeks.

Kalido also implements model changes at the conceptual level. The business representatives review the first reports produced from the warehouse and, working on their existing model together with the Kalido Architect, they decide on changes to improve the model. Changes, which might include adding new attributes and measures, adding a missing source transaction, an additional hierarchy or a new association type, are made to the business model itself. All these are defined graphically in the Kalido Modeler. At this point, the new version of the model is “re-deployed” to the warehouse. Kalido will understand the model changes and automatically modify the warehouse schemas without any SQL coding or interception by developers or DBAs. Sometimes, the changes involve only modified reference data, such as alternative classifications of products, an alternative internal organization or a changed business rule table. After a couple of iterations, each taking 1-2 weeks, we typically reach a state that satisfies business representatives and therefore enables us to start productizing the BI development.

At this stage, Kalido Universal Information Director automatically generates BI conceptual layer components by exporting a subset of the business model into the BI layer. This bridge provides all semantic definitions of the business metadata with SQL mappings to the Kalido warehouse reporting schema tables, thereby delivering a consistent, accurate and complete source to BI. Then the development of end user reports can start.

The transfer of metadata components to the BI layer is not a one-off solution. Any changes, scope extensions during the following iterations or after the first go-live, are detected by the Kalido Universal Information Director and incrementally applied to the BI metadata tools. From end-to-end, from start-
to-finish, through iterations and during the whole life cycle of the DW/BI architecture; there is no disconnection.

The biggest impact of Kalido is seen on the data warehouse design, development and maintenance process (see Figure 3). The data architect is empowered as he/she is responsible for not only the initial design of the data warehouse, but the full life cycle. This responsibility is shared with the business representatives, who gradually take over the ownership of the content of the business model. The data architect becomes an information architect. Through this process, we eliminate disconnections. The system-user interfaces are reduced; all user interfaces are presented at the business model layer. In addition, the business model drives all software functions. The data warehouse immediately reflects what is needed by the business and quickly responds to change requests. Business understands their data warehouse content and capabilities because it is represented in business terms in their model. The end-to-end architecture is semantically integrated.

Avoiding the Business-IT Gap with Kalido Data Warehousing

So how does this impact the disconnects we discussed previously?

The definition of business requirements is both easier and more understandable with a visual model. A picture is worth a thousand words, and having a picture with business terms is even better. A business person will easily understand this model. Printing this model and pinning it on the wall enables users to visualize their sales volume measure from the Sales transactions and the one in the Monthly Targets transactions and visually follow them through associations to the common Classes where comparisons can be made.

The Architect is also happy, not only because they are on the same page with the business representative, but also because the IT level details, like cardinalities and the data types of attributes are all captured in the tool.
End-to-end flexibility and the fact that any model change is applied as an incremental metadata extension grant the comfort to business representatives that they are not forced to perfectly define the requirements in the first round. They can discuss and change them later, iteration by iteration. The isolation of the physical schema from the business model and the immediacy of the multiple physical schema generation and maintenance makes those iterations very rapid. We eliminate Disconnect #1.

**Agreement on the model:** Project teams have the luxury of gradually changing the model and improving it as agreement and detailed understanding are progressed. The Modeler helps to make sure the model is implementable via real-time validation. Instead of a logical model, you get a business model in business terms. Even better, what you see there is what will be implemented in the warehouse. The architect and other technical team members are also happy because all the data warehousing best practices are automatically implemented. As a result, we eliminate disconnect #2.

**Understanding the physical design:** The physical design and schema creation is fully automated. Yet, the schemas for staging, storage and reporting are fully “open.” The ETL team can see the staging area as their target, and they are only required to bring the data into the Kalido managed staging schema. Warehouse reporting schema and data mart schemas are open through many API layers, so for BI developers, it is a single and open source of integrated data. Both the ETL and BI layers can see the warehouse structure at the metadata level. We eliminate disconnect #3.

**Editing physical design in the DBMS** is unnecessary because the underlying physical schema of the generic storage area is created during the Kalido installation and is always stable, for any model, for any industry and for any scope. The warehouse reporting schema is generated with scalability and tuning best practices by default. However, as the warehouse grows and usage patterns emerge, DBAs are still able to make performance tuning changes through additional indexes, partitioning etc. We eliminate disconnect #4.

**Schema changes during development:** Changes due to technical reasons or business requests are made through the Modeler and re-deployed automatically as delta imports. We eliminate disconnect #5.

**Model-Warehouse synchronization:** The model is always in synch with what is actually built in the warehouse. The Architect and his/her Modeler never fall off-line. The model shows what is built in the warehouse, so business representatives understand the warehouse. We eliminate disconnect #6.

**Semantic definitions** are made once and carried over all the way into the BI. We eliminate disconnect #7.

**Master Data consistency:** Master data structures are defined in the model and the management of the data is driven by the Kalido MDM tool. The user is empowered and in control of processes for the change, validation and authorization built in through workflow processes. We eliminate disconnect #8.

**BI consistency:** The Kalido Universal Information Director automatically creates the object definitions in the BI layer. That Kalido bridge then feeds BI tools with data from the integrated warehouse reporting schema and data marts, so that the BI tools remains consistent with the warehouse. We eliminate disconnect #9.

**Responsiveness to business changes:** Business changes after implementation are reflected first in the Modeler and then followed through to implementation through development user acceptance testing and implementation. Major changes, like a significant scope extension or due to a merger/acquisition, follow the same route. We eliminate disconnect #10.
## Disconnects

<table>
<thead>
<tr>
<th>Disconnect</th>
<th>Traditional design and development</th>
<th>Kalido Business-Model-Driven Data Warehousing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition of Business Requirements</td>
<td>Mostly as text document. Business is forced to define 100% accurate definitions</td>
<td>Iteratively developed during the development cycle and visually documented with the Modeler.</td>
</tr>
<tr>
<td>Agreement on the Logical Model</td>
<td>Logical data models are hardly understandable by Business</td>
<td>No need to create a logical model. A data warehouse implementation specialist will just add some warehouse-specific objects to the business model.</td>
</tr>
<tr>
<td>Understanding the Physical Design</td>
<td>Business representatives don’t understand the physical schema designs. Architect creates a database creation script and hands over to the development team</td>
<td>Not needed. Business representatives never face the physical model; The data warehouse implementation specialist does not need any SQL coding.</td>
</tr>
<tr>
<td>Editing physical design in DBMS</td>
<td>Generally needed by DBAs for fine tuning before the database creation</td>
<td>DBAs are still able to do performance tuning on the warehouse schema without changing the design</td>
</tr>
<tr>
<td>Schema changes during development</td>
<td>Done at physical schema, because the data model is not equivalent to as-built databases and not all data model changes can be translated to Delta DDL</td>
<td>Defined in the Kalido Business Information Modeler at the business model level and automatically applied to the warehouse as delta import</td>
</tr>
<tr>
<td>Model-Warehouse synchronization</td>
<td>Graphical data modeling tool is disconnected. Not maintained after the initial design</td>
<td>The graphical model as defined in the Kalido Modeler is always integrated to the Kalido data warehouse, master data and BI definitions throughout the full life cycle</td>
</tr>
<tr>
<td>Semantic definitions</td>
<td>Requires manual effort to pass to BI through the DW physical schema</td>
<td>Carried over from Kalido Modeler to Kalido data warehouse, then optionally to Kalido MDM and the BI layer as a part of automated meta data interfaces</td>
</tr>
<tr>
<td>Master data consistency</td>
<td>No MDM tools in support of DW/BI. Some data quality tools are used in batch mode</td>
<td>Provided by Kalido MDM</td>
</tr>
<tr>
<td>BI Consistency</td>
<td>Fragmented object definitions between multiple BI Conceptual Layer components or embedded SQL in reports</td>
<td>BI object definitions are auto-generated from the warehouse and maintained with live bridges</td>
</tr>
<tr>
<td>Responsiveness to business changes</td>
<td>Each change requires 2-6 weeks and specialist professionals</td>
<td>Led jointly by the Kalido Architect and business representatives and done in days or even hours</td>
</tr>
</tbody>
</table>

Comparison of disconnects: Summary of how business-model-driven data warehousing avoids the disconnects that result from the Business-IT gap in conventional methods.

In short, the warehouse continues to support the business as it changes, thereby improving business agility. With all these capabilities, a Kalido data warehouse lives much longer than conventional ones.
Benefits of Business-Model-Driven Data Warehousing with Kalido

By storing the business model as data and knowing that the data within the data warehouse is always consistent with the model, Kalido’s patented approach is able to provide the following dramatic enterprise data warehousing benefits:

- Improved communication with business people as discussions can center around objects they understand rather than technical constructs of the IT world.
- Iterative and rapid creation of production data warehouses – typically within 8-12 weeks (as compared to the 16 months typically required to build an enterprise data warehouse).
- Rapid responsiveness to changing business requirements and conditions.
- Reduced annual manpower ownership costs of Kalido data warehouses – usually less than half that of traditional enterprise data warehouses due to the speed and ease with which functionality is changed and extended.
- Automatic preservation of historic context for trend analysis and audit reporting.
- Improved regulatory compliance through ensured data traceability (because it only allows information that conforms to the business model to be included in the data warehouse). The Kalido warehouse also maintains data in its true historical context with detailed logs of all processes, ensuring accurate data lineage and true referential integrity at all levels.
- Ability to implement even very complex types of tabular business rules, like business segmentation to support management and measurement of highly matrixed organizations.
- Ability to define corporate standards, such as those desired for product categories, profit calculation and market segments, etc., while enabling local business unit autonomy and variation to co-exist.
- Ability to view information from different perspectives (e.g., by country, by brand, by customer, etc.) based upon the same underlying data: A single version of the truth presented in many different contexts. Gone are the drawn out discussions about whose figures are correct and the days where the sum of each division’s profit adds up to more than the company total.
For more information on Kalido

If you would like to learn more about business modeling or Kalido, please visit www.kalido.com to read other white papers and to see Kalido product demos. You can also contact Kalido and request a meeting to discuss the principles of business-model-driven data warehousing and find out whether you can benefit from it. We also encourage you to evaluate Kalido’s enterprise-scale Master Data Management application (Kalido MDM), which is developed based upon the same generic modeling and adaptive data store principles.

About the Kalido Information Engine

The Kalido Information Engine is a suite of products that gives customers the ability to manage their entire BI infrastructure. Unlike traditional approaches, the Kalido Information Engine puts information management in the hands of the business, while satisfying the strict data management requirements of IT. Working in unison with existing transaction systems, data warehouses and front-end BI tools, the Kalido Information Engine brings new levels of flexibility and insight to corporate decision-making that change how, and the cost at which, business gets done.

The Kalido Information Engine is composed of the Kalido suite of products: Kalido Business Information Modeler, Kalido Dynamic Information Warehouse, Kalido Master Data Management and Kalido Universal Information Director™.

About Kalido

Kalido delivers active information management for business. With Kalido’s unique business model-driven technology, decisions are fueled by accurate, accessible and consistent information, delivered in real time, to dramatically improve corporate performance. Kalido software can be deployed at a fraction of the time and cost of traditional information management methods.

Kalido software is installed at over 250 locations in more than 100 countries with market leading companies.

Contact Information

US Tel: +1 781 202 3290
Eur Tel: +44 (0)845 224 1236
Email: chris.johnson@kalido.com

or visit our website at
www.kalido.com