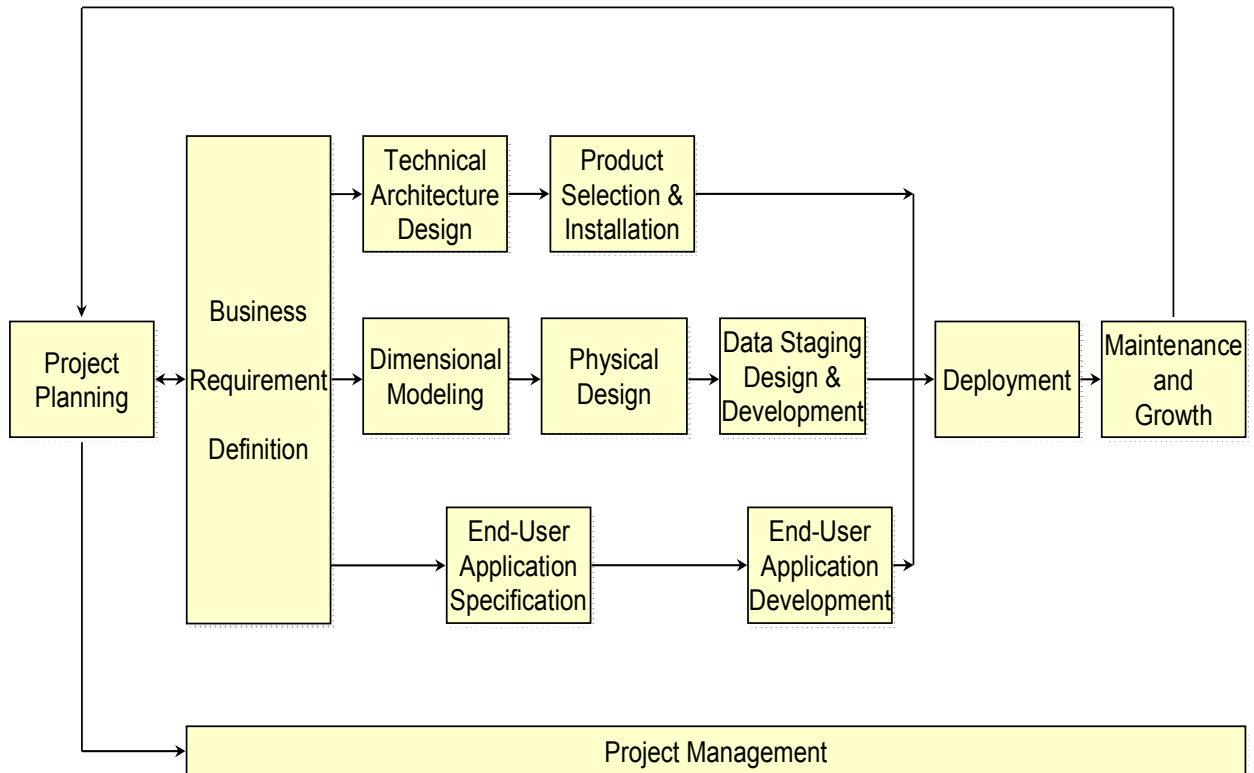


Data Warehouse Lifecycle in Depth
By Warren Thornthwaite (Ralph Kimball Group)
October 2003

Lecture Note Taken by Arman Kanooni

The Business Dimensional Lifecycle



Political Issues in Data Warehouse

Example in Universities:

- Distributed power structure.
- There is no real bottom line.
- Every professor has his own income sources and each one is like his own CEO.
- In order to find how is in charge look for how uses the biggest word!

Having sponsorship helps to get the needed resources as well as required training, and collaboration of line of businesses. Good sponsorship makes a huge difference in success of a Data Warehouse project.

Data Warehouse people are in odd position, i.e., they are half way in business and half way in Information System world. In some companies, business people and IS people do not get along. Management should highlight that problem and address it. To make business people interested and engaged, we start with **Prioritization Process**. We use two by two matrixes.

For example, we start to gather requirements, a user might say we need Revenue Information and we agree on that. Next day, they come back and ask us that “It is really great if we have Cost information as well!” Now, we do have two choices: either we can say “No”! Or try conveying that message by driving them to this answer! You don’t want to say “No” a lot in a Data Warehouse project.

“We can discuss the possibilities to bring Revenue, Cost, and may be Customer to do create additional business analysis capabilities. Then, we have to explain for example Cost data is coming from 4 different systems and not only one system. This will be a lot harder that we have to do. Originally, we all agree that it takes 6 month to do this project and now we are extending it by 3 or 6 more months.”

We lead them to the problem at hand and probably the need for 2 more people.

“I don’t have a head count; do you have a head count?”

It is really about educating the end user. It is important to have this kind of relationship with business people.

Scope Creep

The best way to deal with scope X deliver on time Y being change to X+3 on Y+3 is to build this into the plan knowingly that this will happen. If not, it is an educational process. For example, they told us the source system is clean and now you find out that it is not! Then we (Business and IT) need to think what this means. Data Warehouse will take longer than anticipated or change the scope of Data Warehouse. We have to do one or another. It is not our fault. Somebody lay to us!

It is all depend on relationship built up to this point. And if we have strong working partnership though the education, it is possible to change the scope and requirements.

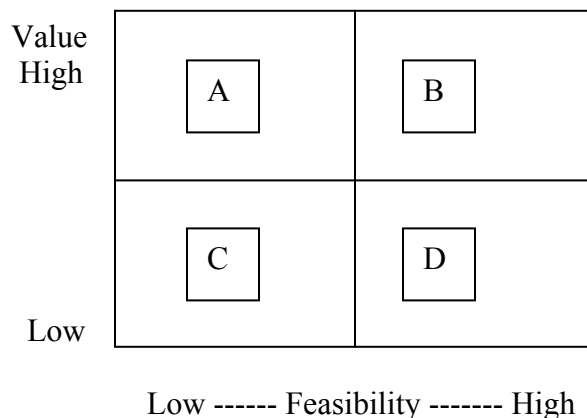
We start to build these relationships when we start the **interviews process**. We get a lot out of it. We get the content. This is almost an art. It is a tool to create credibility. It is always easy to show them a file layout and ask them which one you need! What data we have available? What do you need? What field would you like? The answer: All of them! Then we go back and tell that Users don't know what they want!

During the Interview process, we have to go as Businessperson and let them talk about what they do? For example, ask "What problem you try to solve?" So, the first think we get is the foundation of that relationship. The first time we come in, business people think "Oh, No! Another IS Interview, more Information System people!"

But, if you come and you know what they do in general, what the business issues are, what the competitive issues are, you have a sense for what's going on business world and you know the terms from their perspective. You can talk to them in their language; they start to sense that this is different. We use the "we" word dealing with business people. "How can we solve this problem?"

And if you have done your homework and talk about business and conduct yourself in professional manner: taking note, sending request, etc. It will set the tone for Data Warehouse team.

End result, we get the business requirements, the partnership, and support for Data Warehouse effort. The next step is this process is to build Value versus Effort Matrix.



The Data warehouse project should focus on high business impact/high feasibility themes initially.

- **Theme B is the place to start.**
- Theme A: High Impact but Impossible to implement.
- Theme D: Easy to Implement, but not very valuable to organization.
- Theme C: AVOID as impossible to implement and useless to organization.

This is a classic business tool. This is entirely centered on business. There are 2 sides to this matrix. They are Value and Effort or Feasibility. We gather "Themes" or "Business

Area” during the interview and evaluate their business value and level of effort to achieve it. If we have done the interview process right, those themes should be evident to business people. We deal with those themes in Data Warehouse prioritization process. So we are constantly building credibility in Business organization.

Dealing with requirement around organization unit such as sales group does not help from data perspective. For example, sales group might be interested about all sales data and also might want to know about the inventory, because they want to talk to their customer what is available or not. They might be interested in Return. They might want to do Customer Relationships analysis or Product Contribution. They want to know what product they should be selling. They actually group data from cross organization. **So by thinking that we can minimize the scope of first Data Warehouse project by focusing on a single group will not work!** This same group wants every thing. So we take this approach aside and **let’s look at what business processes are evolved.** And find out **which one is useful to all group.** In other word, which one has the highest value? So at the end of this process every one gets value out of it. Then the Data Warehouse gets more value since the data content is valuable. So matrix gives us what are the major business object will participate in our business (customer processes, people, etc.).

Analyzing a business process means understanding its data content (stable part) and not the dynamic view of a process, which can change any time!

Technical Architecture Design

What's the problem?

- Most data warehouse projects don't take the time up-front to create a viable technical architecture. This leads to:
- Inflexible design – unable to meet future needs
- Mid-project surprises
- Missing functionality
- Extra work.

The architecture is based on business problems that it is trying to solve. The aerospace business is about system integration and creating things. You have to have a framework, otherwise it won't work.

For example, building a house is a complex project. There are lots of people involved. In construction we talked about infrastructure, real pipes. Look at a construction site. From a bare land to first occupancy it might take 9 months. Construction people know how to do this. The key is the **blueprint**. A set of drawings and descriptions that include:

- Multiple levels of detail
- Different models for different systems (electrical, plumbing, HVAC, communications, vacuum)
- Standards – plugs, lights, plumbing fixtures, doors, etc.

They start with high-level graphical representation such as side view, plan view, elevation and then we get to the lower level of detail (the 1st floor, the 2nd floor). Then more details about plumbing or window frames, etc are provided for a team of competent professionals can look at these things and go and build it. That's we are working for in DW architecture.

We need enough of detail information to understand what we are doing in DW project. At that point is about the building the thing. What function we need? Where they live? How they work together. We move behind the business requirement and build the technology to support those requirements. We capture those requirements in the design. DW within different companies had built different things, because all they have different business requirements. For example, one of my top priorities in my business might be to help the sale force to communicate effectively with customers about our business relationships.

For example, in a Business-To-Business case where we do lots of business with retailers and we want the sales force to be able to show the product line and how we handle orders. I have lots of sale person needing accessing data in a disconnected fashion. I have to think about this need and how to translate this locally with an engine to distribute data and I need to keep history and having a user friendly front-end to navigate this with customer on site. You see this requirement what I really need to build.

Another company building another Data Warehouse might not have this requirement. All this infrastructure and architecture is not important to them at all. They had gone build fundamentally different DW, same concept different thing. **So, understanding the requirement is really the root of Data Warehouse architecture.** This is similar to the architecture of a house based on the needs of a family wanting to live there. The only difference is that they do have a lot more standard defined and lots of details compare to Data Warehouse architecture.

For example, there is ANSI standard for the door of a garage! I can tell you that we don't have that. The Data Warehouse problems are complex and if we try to simplify them, it will lose meaning. In another word you no longer represent the business. You have to keep a level of detail in your specifications to represent the business. Your business will be harder to capture compare to grocery business for example. In grocery business you bring stuff in and sell to people. Another complex domain is healthcare. You have to go through a triage process to manage the complexity and decompose it in manageable parts so you can talk it based on their priority.

Blueprints are great. They help us to communicate and think through the issues early rather than later. Architecture shows what is gone to our data warehouse to solve the business problem that we needed to solve. What are the functions that we need? How they work together? How we have to maintain it over time? How we expect it to evolve? As technology grow and as business change what our strategy will be for working these questions.

Architecture looks like a graphical representation. It has high-level model. This is the picture of Data Warehouse basically. This is the picture of sub systems and how they fit together. Then it goes to the detail. We write out the functionality we need. For example, we need local data mirroring capability. We need clear level of detail that when we go build it we have enough information.

How this map to Zachman framework? He said that in IT we are late, over budget and said who does it well? Basically, he took at aerospace industry to create his framework. We took Zachman framework and adapt it to Data Warehouse. In this case, we have **Data Architecture, Technical Architecture, Application Architecture and Infrastructure.**

This framework helps to realize what component we need to build the thing. We don't want be lost in an architecture. We want an optimal architecture (80/20 rule) in order to save us lots of effort. I don't want make it a career.

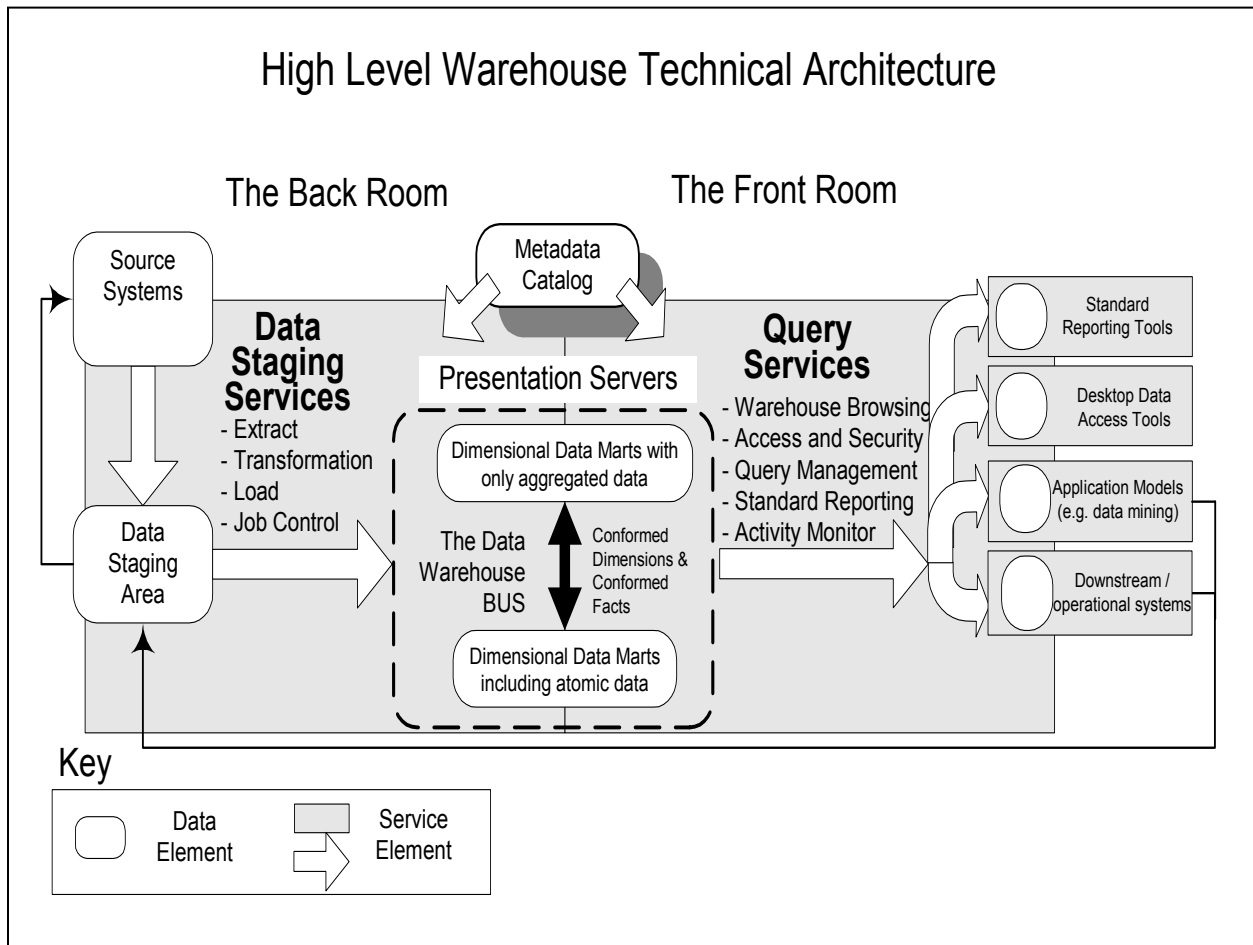
What goes into a typical Warehouse Architecture?

Staging area:

We need get the data from source system and clean it.

There is a user mode coming into DW: Ad-hoc or standard reporting.

You can have a system polling data out of Data Warehouse. For example, our customer care system might need action history out of Data Warehouse. So when somebody call and we do a lookup and actually system get the phone number out of switch when you call and we use that to find the history out of the Data Warehouse and populate one of the page of customer service screen. Each of these user modes has implications for functionality and service level.



Architecture Framework

Data Architecture Area

As stated above, business processes drive the data architecture. For example, in a manufacturing environment the data model might include orders, shipping, and billing. Each area draws on a different set of dimensions. But where dimensions intersect in the data model the definitions have to be the same—the same customer who buys is the same that builds. So data items should have a common structure and content, and involve a single process to create and maintain.

Organizations often ask how data should be represented in the warehouse—entity/relationship or dimensional? “If you have a **star schema**, then use dimensional. Is your detail **normalized** or dimensional? Will users be querying detail? Then use dimensional.” He adds that most data warehousing experts are in substantial agreement; the [data] sources are typically entity/relationship models and the front end is a dimensional model. The only issue is where you draw the line between the warehouse itself and the data staging area.

As you work through the architecture and present data to your users, tool choices will be made, but many choices will disappear, as the requirements are set. For example, product capabilities are beginning to merge, like MOLAP and ROLAP.

“MOLAP is okay if you stay within the cube you've built. It's fast and allows for flexible querying—within the confines of the cube.” Its weaknesses are size (overall and within a dimension), design constraints (limited by the cube structure), and the need for a proprietary database.

Data Architecture Check List

- ❑ Business Requirements and Audit
 - ❑ What information do we need to make better business decisions?
 - ❑ What data assets are available?
- ❑ Architecture Models and Documents - The Dimensional Model
 - ❑ What are the major entities (the facts and dimensions) that make up this information?
 - ❑ How do they relate to each other?
 - ❑ How should these entities be structured?
- ❑ Detailed Models and Specs - The Logical and Physical Models
 - ❑ What are the individual elements, their definitions, domains, and rules for derivation?
 - ❑ What are the sources and how do they map to the targets?
- ❑ Implementation
 - ❑ Create and document the databases, indexes, backup procedures, etc.

Back Room Architecture Area

The technical architecture is driven by the metadata catalog. Everything should be metadata-driven. The services should draw the needed parameters from tables, rather than hard-coding them. An important component of technical architecture is the data staging process, which covers five major areas:

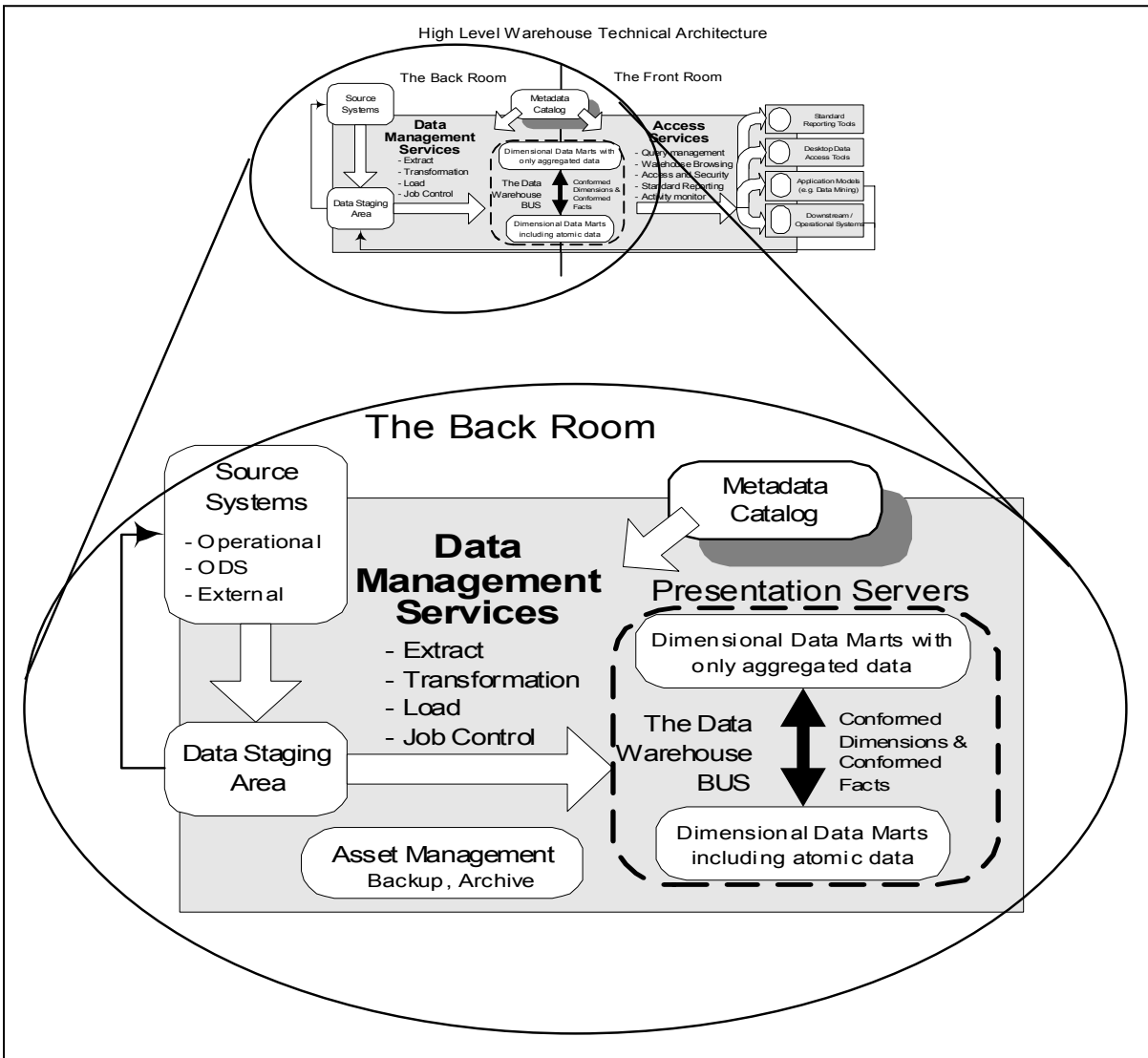
- *Extract* - data comes from multiple sources and is of multiple types. Data compression and encryption handling must be considered at this area, if it applies.
- *Transform* - data transformation includes surrogate key management, integration, de-normalization, cleansing, conversion, aggregation, and auditing.
- *Load* - loading is often done to multiple targets, with load optimization and support for the entire load cycle.
- *Security* - administrator access and data encryption policies.
- *Job control* - this includes job definition, job scheduling (time and event), monitoring, logging, exception handling, error handling, and notification.

The staging box needs to be able to extract data from multiple sources, like MVS, Oracle, VM, and others, so be specific when you choose your products. It must handle data compression and encryption, transformation, loading (possibly to multiple targets), and security (at the front end this is challenging). In addition, the staging activities need to be automated. Many vendors' offerings do different things, so he advises that most organizations will need to use multiple products.

A system for monitoring data warehouse use is valuable for capturing queries and tracking usage, and performance tuning is also helpful. Performance optimization includes cost estimation through a “governor” tool, and should include ad hoc query scheduling. Middleware can provide query management services. Tools for all of these and other related tasks are available for the front end, for server-based query management, and for data from multiple sources. Tools are also available for reporting, connectivity, and infrastructure management. Finally, the data access piece should include reporting services (such as publish and subscribe), a report library, a scheduler, and a distribution manager.

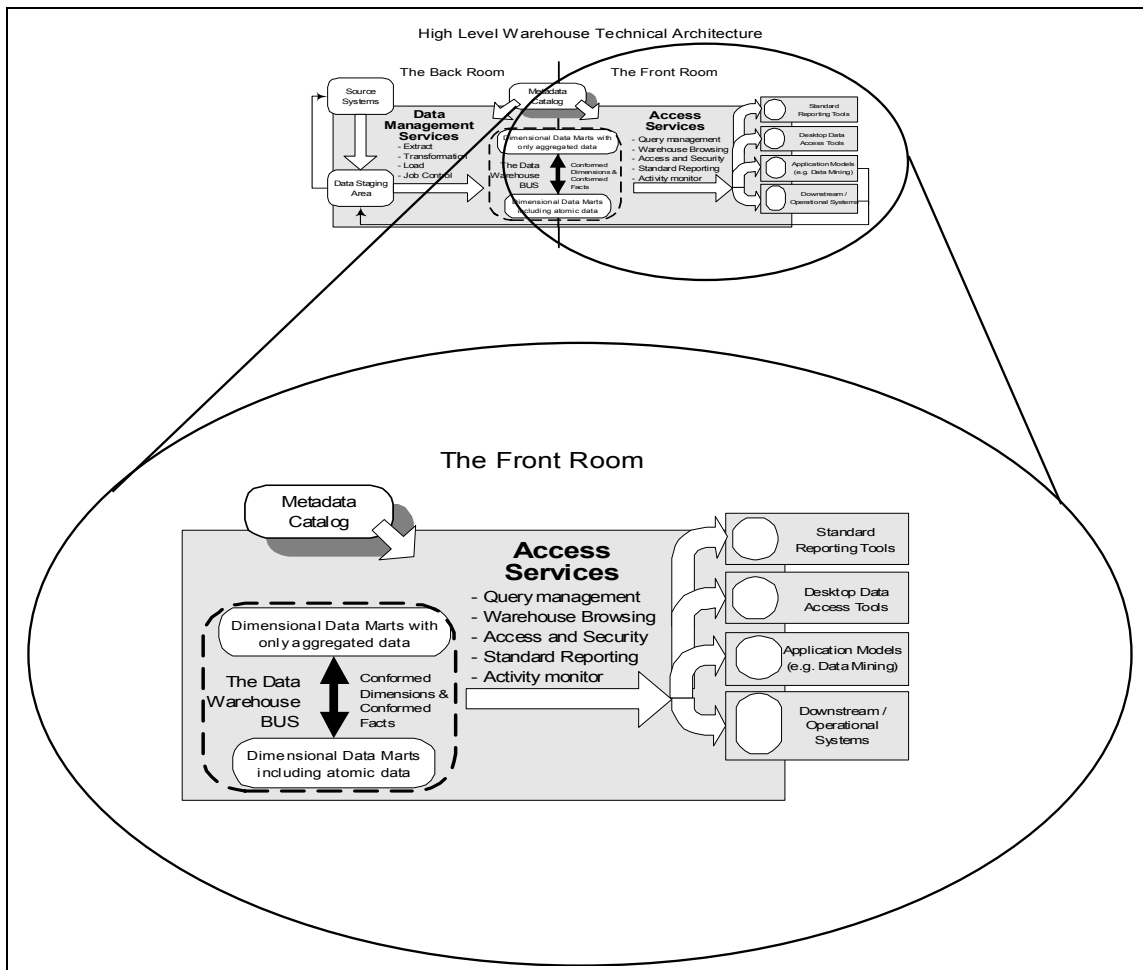
- ❑ Business Requirements and Audit
 - ❑ How will we get at the data, transform it, and make it available to our users?
 - ❑ How is this done today?
- ❑ Architecture Models and Documents
 - ❑ What are the specific capabilities needed to get the data into a usable form in the desired locations at the appropriate times?
 - ❑ What are the major data stores and where should they be located?
- ❑ Detailed Models and Specs
 - ❑ What standards and products provide the needed capabilities?
 - ❑ How will we hook them together?
 - ❑ What are our development standards for code management, naming, etc.?
- ❑ Implementation

- ❑ Write the extracts and loads.
- ❑ Automate the process.
- ❑ Document the process.



Front Room Architecture Area

- ❑ Business Requirements and Audit
 - ❑ What are the major business issues we face?
 - ❑ How will we measure these issues?
 - ❑ How do we want to analyze the data?
- ❑ Architecture Models and Documents
 - ❑ What will users need to get the information out in a usable form?
 - ❑ What major classes of analysis and reporting do we need to provide?
 - ❑ What are the priorities?
- ❑ Detailed Models and Specs
 - ❑ What are the specifics for the report templates?
 - ❑ Rows, columns, sections, headers, filter and so on?
 - ❑ Who needs them? How often? How do we distribute them?
- ❑ Implementation
 - ❑ Implement the reporting and analysis environment.
 - ❑ Build the initial report set.
 - ❑ Train the users. Document.



Infrastructure Architecture Area

With the required hardware platform and boxes, sometimes the data warehouse becomes its own IS shop. Indeed, there are lots of “boxes” in data warehousing, mostly used for databases and application servers.

The issues with hardware and DBMS choices are size, scalability, and flexibility. In about 80 percent of data warehousing projects this isn't difficult; most businesses can get enough power to handle their needs.

In terms of the network, check the data sources, the warehouse staging area, and everything in between to ensure there's enough bandwidth to move data around. On the desktop, run the tools and actually get some data through them to determine if there's enough power for retrieval. Sometimes the problem is simply with the machine, and the desktops must be powerful enough to run current-generation access tools. Also, don't forget to implement a software distribution mechanism.

Infrastructure Architecture Checklist

- ❑ Business Requirements and Audit
 - ❑ What hardware and system level capabilities do we need to be successful?
 - ❑ What do we currently have in place?
- ❑ Architecture Models and Documents
 - ❑ Where is the data coming from and going to?
 - ❑ Do we have enough calculation and storage capacity?
 - ❑ What are the specific capabilities we are counting on?
 - ❑ Do they exist?
 - ❑ Who is responsible for them?
- ❑ Detailed Models and Specs
 - ❑ How do we interact with these capabilities?
 - ❑ What are the system utilities, calls, APIs, etc.?
- ❑ Implementation
 - ❑ Install and test new infrastructure components.
 - ❑ Connect the sources to the targets to the desktop.
 - ❑ Document.

A Word about Meta Data

The creation and management of data has the following “steps” in the data warehouse process:

1. Warehouse model
2. Source definitions
3. Table definitions
4. Source-to-target maps
5. Map and transformation information
6. Physical information (table spaces, etc.)
7. **Extracted data**
8. **Transformed data**
9. Load statistics
10. Business descriptions
11. Query requests
12. **The data itself**
13. Query statistics

To show how important metadata is, look at the steps listed above: only three involve “real” data—7, 8, and 12. Everything else is Meta Data, and the whole data warehouse process relies on it. The major technical elements of a metadata catalog include:

- *Business rules* - includes definitions, derivations, related items, validation, and hierarchy information (versions, dates, etc.).
- *Movement/transformation information* - source/destination information, as well as DDL (data types, names, etc.).
- *Operations information* - data load job schedules, dependencies, notification, and reliability information (such as host redirects and load balancing).
- *Tool-specific information* - graphic display information and special function support.
- *Security rules* - authentication and authorization.

Developing an Architecture

When you develop the technical architecture model, draft the architecture requirements document first. Next to each business requirement write down its architecture implications. Group these implications according to architecture areas (remote access, staging, data access tools, etc.) Understand how it fits in with the other areas. Capture the definition of the area and its contents. Then refine and document the model.

Developing data warehouse architecture is difficult, and we are thus warns against using a “just do it” approach, which also called “architecture light.” But the Zachman framework is more than what most organizations need for data warehousing, so it is recommended a reasonable compromise consisting of a four-layer process: business requirements, technical architecture, standards, and products.

Business requirements essentially drive the architecture, so talk to business managers, analysts, and power users. From your interviews look for major business issues, as well as indicators of business strategy, direction, frustrations, business processes, timing, availability, and performance expectations. Document everything well.

From an IT perspective, talk to existing data warehouse/DSS support staff, OLTP application groups, and DBAs; as well as networking, OS, and desktop support staff. Also speak with architecture and planning professionals. Here you want to get their opinions on data warehousing considerations from the IT viewpoint. Learn if there is architecture documents, IT principles, standards statements, organizational power centers, etc.

Not many standards exist for data warehousing, but there are standards for a lot of the components. The following are some to keep in mind:

- *Middleware* - ODBC, OLE, OLE DB, DCE, ORBs, and JDBC.
- *Data base connectivity* - ODBC, JDBC, OLE DB, and others.
- *Data management* - ANSI SQL and FTP.
- *Network access* - DCE, DNS, and LDAP.

Regardless of what standards they support, major data warehousing tools are metadata-driven. However, they don't often share metadata with each other and vary in terms of openness. “So research and shop for tools carefully”. “The architecture is your guide. And use IT advisory firms, like Gartner Group, META Group, Giga, etc.”

How detailed does a data warehouse architecture need to be? The question to ask is this: Is this enough information to allow a competent team to build a warehouse that meets the needs of the business? As for how long it will take, the architecture effort will grow exponentially as more people are added for its development (i.e., it becomes “technopolitically complex”), and more complex the resulting system needs to be (i.e., “functionally complex”).

Like almost everything in data warehousing, an iterative process is best. You can't do it all at once because it's too big—and the business won't wait. Also the data warehouse market isn't complete yet. So begin with high leverage, high-value parts of the process. Then, use your success to make a case for additional phases.

Conclusions

To sum up, the benefits of having Data Warehouse architecture are as follows:

- *Provides an organizing framework* - the architecture draws the lines on the map in terms of what the individual components are, how they fit together, who owns what parts, and priorities.
- *Improved flexibility and maintenance* - allows you to quickly add new data sources, interface standards allow plug and play, and the model and metadata allow impact analysis and single-point changes.
- *Faster development and reuse* - warehouse developers are better able to understand the data warehouse process, data base contents, and business rules more quickly.
- *Management and communications tool* - define and communicate direction and scope to set expectations, identify roles and responsibilities, and communicate requirements to vendors.
- *Coordinate parallel efforts* - multiple, relatively independent efforts have a chance to converge successfully. Also, data marts without architecture become the stovepipes of tomorrow.

It is recommended that companies align with business requirements but to be practical. He also emphasizes the importance of keeping up with advances in the data warehouse industry. Finally, remember that there is always architecture: implicit or explicit, “almost in time” or planned. Experience shows that the planned and explicit ones have a better chance of succeeding.