Enterprise-Level Architectures

Information is power in the modern world, and organizations with the most accurate and readily accessible data make the fastest decisions with the least negative impact. Making the best business decisions will positively affect the bottom line. This is something all businesses strive for. It translates into competitive advantage for the companies that are willing to invest in it. Of course, there is an investment, and often there is a slow initial start-up time, but it tends to accelerate once the initial setup activities have been accomplished.

In the next chapter you will learn more about the development of the organizational structures, objects, methods, and resources for implementing an enterprise-level information architecture, including the subarchitecture’s enterprise-level system architectures, enterprise-level data architectures, and enterprise-level technology architectures.

Practice

- System architectures ensure that the current and future processing capabilities of the enterprise are not impaired during the development process.
- Technology architectures ensure that the enterprise is developing the right applications on the right platforms to maintain the competitive edge.
- Data architectures are the heart of business functionality. Given the proper data architecture, all possible functions can be completed within the enterprise easily and expeditiously.
System Architectures

An enterprise-level system architecture is an inventory mechanism that provides an automatic checklist of applications by function. This, taken in conjunction with an evaluation of each application within a scorecard range, allows strategic sequencing to take place in the mapping of new development applications (replacements) and remedied ones (reengineered). This ensures that development will take place when it is best suited to do so. Indirectly, it provides a matrix of application code to function, and in doing so provides some input into the reusability of the current code. The business systems architecture provides the mapping of current application systems to current data stores. All of these are of critical importance in maintaining control over one of the most expensive resources that the corporation invests in.

Enterprise Data Architectures

Enterprise-level data architectures ensure that the disintegration of integrated data stores is minimized. This ensures that current activity is sustainable while new development can take place. Also, using the same template also ensures that a foundation exists for the implementation of new techniques and technologies. They place tools and methods in relation to one another by virtue of an engineered structure. They also provide a way of quantifying risks and costing for or against implementing a new component of the architecture. In order to achieve the lofty objective of a corporate or an enterprise data architecture, it requires that organizational management address data as the critical resource and asset that it is.

Enterprise Technology Architectures

Enterprise-level technology architectures ensure that the enterprise is developing the right applications on the right platforms to maintain the competitive edge that they are striving for. Precious time in opportunity assessment is not wasted keeping a structure in place that provides a defaulting choice mechanism for each application. Also, the technology architecture provides a road map within each technology platform to ensure that the right tools and development options are utilized. This prevents additional time being spent extricating the application effort from previously experienced pitfalls.

But architectures aren't enough to ensure that the process and templates are used properly. Without the infrastructure mechanisms
in place, the architectures, processes, standards, procedures, best practices, and guidelines fall by the wayside. We will cover in detail in the next chapter what groups are necessary and what roles they perform. With these data infrastructure mechanisms in place, the architectures have a chance of surviving the onslaught of the chaos brought about by changing priorities, strategic advantage, and just plain emergencies. We will cover the system and technology architectures with more detail in subsequent chapters, where they are more appropriately addressed.

Enterprise Architecture
Terminology—Business Terms

We should take a moment to discuss some terminology and title structures to ensure that we understand those things that are involved in the infrastructure mechanisms. Detailed in the next few paragraphs are some of the terms and objects that we will be talking about.

First is a Business Entity Cluster (BEC). This is analogous to an Alexandrine “center” (as mentioned in the last chapter); a BEC is a consolidation or coalescence of data foci that deal with a “common” area of business subject matter within the corporation. Often business entity clusters appear to align themselves parallel to the abstracted division-level data needs of a corporation. While this is not a requirement, the situation often falls into place that way because it makes sound business sense. BECs are often expanded to the level necessary to cover all data foci in the enterprise's concerned applications that have been integrated or will be integrated.

Within these BECs are groupings or subclusters of entities that are denoted subject areas. It is of great importance to understand that this subject area orientation is concerned with the abstracted views of data independent of any lower process or business needs that are associated with it. They are specific as to data but independent of process. This is to ensure that while it will support the current business activity load, it is open and flexible for future down and outward specification. An example of a subject area would be finance or human resources.

As we descend one more layer we encounter business problem areas, which we can also refer to as data applications areas. These represent the collections of the specific data needed to support the business processes that advance the company's strategies and policies.

From the activity perspective, the Business Activity Segment (BAS) reflects a consolidation or coalescence of business activity
foci within the corporation. Just as in the BECs at this level of abstraction, these are sympathetic in nature to the processes necessary for the organization process needs at the divisional level.

Within these BASs are groupings of subactivities known as functions. Another parallel can be drawn to the data side by realizing that functions are the process equivalent of subject areas in that they represent the abstracted process needs of the corporation at the departmental level. Functions are defined independent and without concern for the lower-level data needs. A function can be defined as an activity that has no start of completion other than with the life cycle of the corporation—for example, accounting or shipping. Again, these tend to resolve themselves to a departmental level of activity.

As we descend one more layer on the process side, as we did with the data, we come to functional process areas (FPAs), which can also be called process applications or simply applications. With this parallel structure in mind between process and data and the organizational levels associated with it, we can start dealing with composite objects and organizations specifically that address them.

The Enterprise Model

A compendium of the highest level of data and process models is an enterprise architecture model. This is a model that captures high-level business entities (BECs) and high-level business processes (BASs) that reflect the major reasons for the enterprise’s (corporation’s) existence. It is highly abstracted in nature and content, and it looks at things from the 50,000-foot level. There is not a great deal of detail, but the main subject areas are defined within their BECs and high-level functions are defined within their BASs. Further, external, subsidiary models deal with the specification of data at the application level.

In previous paragraphs we spoke of data architectures and system architectures as being part of the enterprise architecture. Other names we will use as synonyms for the manifest product of these architectures will be corporate data model and corporate activity model.

The Enterprise Data Architecture from a Development Perspective

The major premise here is that the enterprise data architecture (or an active copy of it) will be the source and repository of
all development models. This ensures consistent development, minimization of disintegration, and enterprise data architecture concurrency. In the following paragraphs we will talk about the roles and responsibilities of various levels of management. We will do this in parallel with the different levels of data abstraction.

To do this, we will look at data and organization from a top-to-bottom approach. If we look at the major phases of a model-driven development process, we will see that each of these stages is definable, is discrete, and produces work products that are usable in the next phase of the development process. These stages are an analogous implementation of the Zachman framework stages. The first of these stages is planning.

Planning

Planning is the major function that provides a road map into the future that includes all strategic efforts and the ability to respond to competition-triggered or spontaneous events. By defining that road map into the future, resources can be planned for, expenses can be projected, purchases can be made, and deliveries can be completed on or near the time they are necessary to be completed. Models developed here will feed the analysis and design stage.

Analysis and Design

Analysis is a major function by which business requirements are investigated and documented in such a way as to be reusable for other purposes such as reference, validation, assessment, education, and traceability. By capturing the business requirements for each business area, the processing needs of that area are directly addressable and ensure that the knowledge of the business application is defined. This analysis is done independent of consideration of the organizational structure and the technology platforms available. This is to ensure maximum flexibility in choice for the target architecture. Models developed here will feed the transformation or translation stage.

Transformation

The transformation process translates the business requirements in a logical business model into a model that is “acclimatized” to the target environment it will be operating in. This translation includes DBMS specification as well as resource specification such as physical DASD storage for data and indexes.
Also included in this stage are those changes to the logical and physical model structure that will ensure good performance. The overall effort is to create the smallest physical “footprint” on DASD for the resulting database while still retaining all the original characteristics of the business requirements. Models developed here will feed the implementation stage.

Implementation

The implementation process moves the translated model into a physical environment. This includes the utilities that are run against the database, as well as mechanisms that are created to recover or secure images of the data for security and safety purposes. The implementation also allows active programs to execute their processes against the data store. This is known as the application function and is the only reason that the database exists. The application code or programs allow the business user to interact with his data in a formalized or ad hoc manner.

Practice

The following are the major factors in the success of the implementation of an enterprise architecture:

- Identification of subject area
- Identification of subject area drivers
- Naming and object standards
- A commitment to data sharing
- A data dictionary tool or lexicon
- Defined and controlled domain constraint data
- Proper organizational controls

Subject Area Drivers

Before we go much further, we must consider some facts about subject areas. Within each subject area lies a core or base entity that is the focus of the subject area—the nucleus of the Alexandrine “center.” It is, as it were, the kernel of the subject area. It often is represented by an entity whose primary identifier can be readily correlated to the filing tab hierarchy that was used when the system was manually controlled. If the system wasn’t a manual system, then it would be the file organization key, such
as the key set in a VSAM database. These kernel entities and their identifiers are collectively known as subject area drivers. A few examples of kernel entities in a financial area are loan, account, and customer. In a manufacturing area, they would be product, market, sales, and inventory (both material and product).

This is because when changes are made to these entities, it most often ends up being propagated throughout the entire subject area. Knowing what subject area drivers are critical to an enterprise often allows subject areas to be skeletally defined and then fleshed out over time in the process of enterprise activity modeling. There are critical success factors for subject area implementation when this is done. Aside from the architectural dependencies we have just noted, there are others that are more indirect.

Naming and Object Standards

One of the most critical components of the architecture is a defined set of naming structures for all objects in the process. The identification process helps define which activities are associated with what data items, as well as specification as to what stage the object is in the design process. Object names should be the result of a consistent translation of the business reference to the object assigned in either a manual or automated mechanism that ensures uniqueness. The lexicon or data dictionary for the individual application must reuse the corporate lexicon in order to ensure data sharing opportunities. (A friend of mine in the industry once said, “The biggest problem with data dictionaries is that they are often written by IT people, not businesspeople. They tend to state the obvious (e.g., restate the name) rather than provide any real insight into meaning, domain of values, usage or source.”)

Often there is an adjunct to a passive lexicon or data dictionary. This is an automated routine for the generation of database object names based on the known or defined standard. The names often consist of a root, one or more modifiers, and a class word. The root is the main descriptor of the name and tells the reader what the object is concerned with. The modifiers are qualifiers that amend or further define the root. The class word defines what type or class of object it is. An example would be:

Root = account
Modifiers = overdraft and limit
Class word = code

Therefore, the name would be “account overdraft limit code.” The routines themselves use algorithms to abbreviate the formal, long names that may be up to 30 to 60 characters long into...
something more acceptable to the programmer and DBMS limitations. The abbreviated names are then used in the creation of the database objects.

The initial start of these algorithms is usually a base pool of the appropriate industry abbreviations. If a known abbreviation is found, it is used in the name. If no abbreviation is found, the abbreviation algorithm is engaged to shorten the name. Because the algorithm always functions the same way, the names for similar or related objects have resemblance and consistency.

This algorithm can be used for data names in all stages of development but is most critical in the transformation/translation stage. Having standard names for the same objects ensures that there is consistency among all those involved in the design process when referring to specific objects. For example, developers can talk to database administrators and clients in the same language by using the same object names.

An additional character may be used in the physical names to indicate the object type when there are several that are derived from one. For example, a view of a table may have a “v” in the name at a particular node to indicate a view. An “I” may be used in the same situation to denote indexes.

Data Sharing

In order to accomplish maximum productivity in an enterprise architected environment, it is critical to define those things that are associated with the sharing of data. Among the most critical are the characteristics of the data itself and some of the problems that arise with multiple users of the same data. First, we will cover different data classes, and then we will discuss the sharing rules and limits.

Data Sharing Requirements

1. Data sharing should be defined as a policy and standard approach. In effect, all development must be sanctioned by management as being rooted in the subject areas and that a standard data-driven approach is defined and published by management.

2. Data ownership, data content security, and action sequencing must be resolved. Specifically, the ownership of data must be defined. Initially it must be defined at the entity level and subsequently at the attribute level. Ownership definition includes specification of all create, read, update, and delete (CRUD) categories. Also, data content security must be defined. This
includes the change rules concerning the data content as well as the release/distribution of the data. Finally, action sequencing must be accomplished to ensure that the shared data is accessed at the appropriate time in the attribute life cycle. This action sequencing defines in what sequence the data is updated, changed, or deleted.

3. A glossary of data sharing terms must be available for reference by the users of the data. This is most appropriately addressed by having a complete and comprehensive data dictionary.

4. Naming standards for entities and attributes must foster understanding of the data. The names of the attributes and entities must reflect the real business use of the data. No two attributes can have the same name. There must be one primary agreed-upon name for an attribute, and alias names should be discouraged in the long term.

5. Validation logic and translation rules must be defined for domains being shared. Valid values and ranges must be agreed upon and published for use in accesses of the domain data. Translation rules must be defined to minimize the proliferation of aliases.

6. The shared data model must be the simplest nonredundant image of the data that can be constructed (using canonical synthesis or CASE tools like Er Studio or ERwin).

7. Domain constraint data (valid value, valid ranges, translation, existence, and algorithmically derived data) must be separated from business data with no keyed relationships to the applications data.

8. Generalization hierarchies must be fully expressed (all super type–subtypes defined) in order to ensure that all data are available for sharing. This allows future or shared development to occur.

9. The logical data format must exist in a standard form (minimally, third normal form). This is generally documented in an ERD and associated attribute lists. This is true for structured as well as object approaches.

10. The stability of the business rules concerning the business data must be defined. If the business rules are not defined, the shared databases disintegrate into individual application databases reflecting singular business views.

11. The business need time frame for data sharing must be practical. It is inappropriate to have business users that have different data refresh requirements on the same database unless there is a lowest common denominator that they can share.

When the subject areas are defined, there are logical integration issues that must be addressed to ensure that current and ongoing
activity can be coordinated. Resolution of these issues also ensures data sharing capability. As in all situations where data are gathered for common use, there are some considerations and “rules” that should be observed to maximize the use of the data:

- All subject entities must be added to an existing subject area. If a suitable subject area does not exist, it must be created.
- Project models must be reconciled to the subject area logical model in order to get into the release concept with implementation methods.
- The subject area logical model must remain as close to third normal form as possible. Collapsing and other forms of denormalization should not be done in the SALM but can and should be done in the appropriate application physical model.
- Relationships can exist between subject areas. It is, after all, merely a relationship between two entities within entity clusters. Optionality of relationships should be handled according to prevailing standards on the topic.

Data Dictionary–Metadata Repository

A data dictionary represents a compendium of all data definitions at the lowest level. That is, it consists of data attribute names and the definitions and characteristics associated with them. Normally it is established at the enterprise level but sometimes at the application level on an exception basis. While it is not necessary to compile this, it can be used as a guideline or source of new data names.

The enterprise level lets the pool of data attributes be reused throughout the enterprise, ensuring integrity of output while fostering understanding of the data. While it is critical to have a data dictionary of some kind, it doesn’t matter how it is implemented. As long as it contains or references the procedures and policies that ensure that all development is assisted or implemented by way of a data dictionary, it will ensure success and data sharing.

Dictionary policies and procedures must be defined and publicized due to the need for the developer, the modeler, and the client to all agree on how to encode the requirement in the dictionary. It must be sponsored from IT management as well as client management, since it is often seen by the client as unnecessary overhead. But, as we have seen, once it is defined for the transaction system, it becomes available for the reporting and EIS systems that will follow later on. It will also provide a basis for data sourcing for the data warehouse that will eventually be designed.
Domain Constraints in Corporate and Non-Corporate Data

Domain constraint data fall into two levels of distinction. The first we can refer to as the corporate level; it represents that set of data that the corporation, as a whole, uses. That is to say, it is reference data for all departments in the corporation. This type includes company office tables, zip code tables, shipping tables, department cost codes, as well as other translations, and the like.

The second level of domain constraint data is those that apply to an individual application and represent domain limits for data unique to that application. Examples of these are permitted values for car color in the 1998 model year of General Motors trucks and postal codes for shipping locations for specific product types. Therefore, the second category is at a lower level or more specific level of detail. Whether the domain constraint is first or second level is immaterial when it comes to validation rules/policies and translation rules/policies. These two must be defined to ensure that the domain constraint is used properly and accurately reflects the true limits required by the business entity using it.

Organizational Control Components

The organizational components that engender full control of an architected approach to database design include data administration, database administration, and model repository management. In the context of having a comprehensive strategic data plan, having data architecture implies that the infrastructure of the organization is present and competent to handle the needs of the organization. Therefore, enterprise data architecture must include the mechanisms to support the models of the organization's data:

1. The capture and transformation of logical data models
2. The capture and retention of the physical data models and schemata
3. The process and means by which the physical models and schemas are implemented
4. The DBMS engines and DBMS extensions that will be used to support the architecture
5. The products used to manage the database, such as the tools and techniques that are used to ensure data integrity and quality on the platforms where they are housed

This is a tall order but critical to the success of the effective organization. Let us cover each of these areas in turn.
Data Administration

The data administration area consists of the personnel who are involved in the capturing of the business requirements from the business problem area. Also, they are responsible for integrating with and receiving model constructs and high-level definitions from the corporate architects and capturing these within reusable constructs such as case tools and data dictionary/repositories. They are also responsible for maintaining these model structures over time and ensuring that they reflect the business.

Data administration’s focus is on managing data from a conceptual, DBMS-independent perspective. It coordinates the strategies for information and metadata management by controlling the requirements gathering and modeling functions. Data modeling supports individual application development with tools, methodology(ies), naming standards, and internal modeling consulting. It also provides the upward integration and bridging of disparate application and software package models into the overall data architecture. This overall data architecture is the enterprise data model and is critical in the organization’s ability to assess business risk and the impact of business changes.

Database Administration

The database administration area is responsible for the structures that will be designed from the models that the data administration area produces. Also, as input they will take information about where the application will run and how it will be used in order to structure and organize it appropriately.

As the multitier architectures, data distribution and replication, data warehousing, stored procedures, triggers, and Internet data management bring new focuses to bear in the information processing community, the database administration area must respond to these pressures in a rapid and infrastructurally sound manner. Many organizations, through growth or unmanaged technology architecture, find themselves in the unenviable position of managing and controlling multiple DBMSs with anywhere from two to two hundred databases of each type. Keeping control of an armload of live eels is easier than managing this type of environment.

What can an organization do to counterbalance the entropy that results from these complex environments? What are the main problem areas? The following are some of them:

- Multiple hardware platforms, such as mainframe, server (database, network, and Web), and workstation
• Different operating systems that each have their own command set and interface
• Different DBMS engines that operate from different metamodel architectures and control management languages
• Multiple physical locations that distribute data across the street, town, state, or country
• Middleware connectivity that is used to connect all the different locations and hardware and operating systems
• Data management tools that can be used to move, massage, restructure, propagate, replicate, and maintain large structures housing the different types of data
• Managing the application/DBA support interface, which will provide the efficient development for the many applications that will serve the multitude of users

The depth of knowledge required by the DBA organization in these areas is substantial, depending on the organization’s investment in each area. Because data is the focus in modern information processing, it is the core of the applications and in the applications of the DBMSs. Unfortunately, they are seen as bottlenecks in the process. This is simply because the complexity of the environment allows or promotes performance degradation and the breakdown of processes within it.

How does an organization manage to keep up? In the old days (a few years ago at the current speed of technological evolution), the DBA was a crotchety technologist who had unquestioned technical information and absolute authority over the data. This is not so anymore. Because of the speed of evolution, the technical absolutism has given reign to a conceptual knowledge of the internal structure of the DBMSs and databases. Technical knowledge alone is insufficient to ensure the success of the DBA function within an organization. Today, for example, business rules, relational optimization, access methods, integrity constraints, stored procedures, and user-defined functions exist within the database. The database isn’t an owned thing anymore. It is shared by the DBA, the application, and the user.

The most that can be truly said today is that the DBA area owns the structure of the data and has a custodial responsibility for the data integrity and data quality. What tools does the DBA need in order to function properly? Simply put, they fall into three categories:

1. Object management tools that enable the DBA to perform everyday functions on the objects with his or her domain. Their functionality is limited to object migration, browsing, and modification.
2. **Utilities**, which are the tools that allow the DBA to maintain the databases. These maintenance functions include loading data, unloading data, reorganizing data, backing up data, recovering data, and validating data structure integrity.

3. **Performance monitoring tools**, which are tools that help identify and correct performance problems such as performance monitors, SQL analyzers, capacity planning and performance modeling tools, and systems adjustment tools.

These tools are necessary for the ongoing success of DBAs; make sure you have them on hand.

### Setting Up a Database Administration Group

The most commonly asked question is, “How do I set up and develop a DBA organization?”

Well, the rules are flexible and customizable, but the most common areas to consider are the following ones:

- **Build a centralized DBA area.** In other words, have an area that supports both production and development. This will keep maximum depth of support, foster cross training, increase communication, and provide continuity within the development life cycle (no transitions within the development cycle). It serves to ensure information sharing and creative solutions to major problems. Most crises are solved by the meetings held in the cubicle aisles and not in the conference rooms.

- **Place the DBA area in optimal position within the IT organization.** The DBAs are the custodians and stewards of the data asset for the organization and as such approach database design from a long-term and enterprise-wide data strategy perspective. Their client (application development and maintenance), however, approaches the data from a deadline-oriented, project-driven, and tactical perspective. This is a guaranteed collision that needs to be managed.

- **The DBA area must have significant autonomy in relationship to the client community.** If they do not have it, then their effectiveness is degraded and their expertise degraded to being merely rubber stamps to the application development area’s whims. A truly strategic information resource management function should encompass both data administration and database administration and report at the CIO level. If this is not possible, then it should be a separate peer-level organization within the support groups.
• **Embed continuity of objectives in functional areas.** This continuity must exist among the DBA, the data administration, and model repository management areas. It is critical that these three areas have a seamless and rapid method for design development. If this is not done or is poorly engineered, it is a self-fulfilling prophecy: the design process becomes an impediment to the development process. It is an absolute requirement that these three areas work in tandem or lock-step, with the work products of the first feeding directly into the second and so forth. The policies and procedures should dovetail, and there should be no loose ends to prevent full closure of the design process. This will also provide a complete audit trail from the analysis stage through design and finally to implementation.

• **Publish standards for the development process and the implementation of databases.** These standards cover the naming of database objects, coding of SQL, use of triggers and stored procedures, commit frequency, and referential integrity, among others. This type of documentation should be in the developers’ hands before they create the prototype or proof of concept databases. An education process may be necessary with the developers that allow questions and answers to take place that will allow ambiguity to be resolved. This should also have specific details as to how purchased software packages are handled upon selection.

• **Perform design reviews and preimplementation walkthroughs.** Design reviews should take place with the specific people needed for that level of validation. Architecture and scoping should be done with the user, analysis and design should be done with the application leader and team, and implementation should be done with operations and support organizations.

• **Implement service-level agreements.** The user customer should understand the operational climate of their application and database. Their service level should be defined and published with system documentation to afford future monitoring. The specific metrics that will be used to agree on successful performance should be defined and published. The following categories should also be addressed, such as responsiveness to error call, hours of support coverage, availability, maintenance windows, and recovery time.

• All of these and other specifics will help ensure the success of the database administration process within the IT organization.
Chapter 3  ENTERPRISE-LEVEL DATA ARCHITECTURE PRACTICES

Repository Management Areas and Model Management

The model repository management area is the group of personnel who are engaged in the maintenance and integration of all application models to the model inventory as well as to the corporate model. They are responsible for the maintenance of that data store that encompasses the data dictionary whether it is part of the modeling CASE tool or not.

An enterprise’s information architecture must be capable of containing multiple levels of information (i.e., conceptual/planning models, logical models, and physical design models). The capture of information can be top-down, bottom-up, or middle-out, depending on the tools and methodologies being used.

Many current industry reports support the strategy of using multiple BPR tools. These reports state, “Using direct bridges between multiple modeling tools that have been purchased over time will in effect build a best-of-breed solution for large-scale enterprise modeling and may be the best decision given the amount of investment an enterprise has in a given set of technologies.”

The model management policy must support a release-based system development methodology. A release is a group of business processes that can be delivered with a minimum of time and effort without compromising the options for the delivery of the rest of the business processes.

A model management strategy is of little use unless there are policies and procedures in place that back up the strategy. By this we mean that models that are generated at the application level are seeded from an enterprise model and are reconciled back to it. If there is no enterprise model to source from, it can be built by aggregation. This is a process by which the enterprise is built by integration of all of the modeled application views. Also to be considered is the history of the models, which represents the application requirements state as of a given date.

Also critical in this subcomponent of the infrastructure is the need for training of the user of the model management process. This includes those application personnel as well as repository personnel who are involved with the retrieval, update, and reconciliation of the models to the enterprise or corporate model.

Another area of significant concern is the area of human resources that are to be invested in the process from the user or client community. These subject matter experts (SMEs) are those individuals who have a complete and thorough understanding
of the business processes and the business data. They are critical in the requirements-gathering phase to ensure that the true requirements have been met by the design and also to provide issue resolution when and if this occurs between the application developer, the data administrator, and the database administrator.

In summary, the enterprise-level architecture and model are dependent on the existence and coordination of infrastructure areas that maintain the currency and quality of the enterprise model, the subject area drivers, the data-sharing standards, the data object naming standards, the development and implementation methodology, and the rules that control the domain constraint data.

These infrastructure areas are the information architecture group, the data administration area, the database administration area, and the model repository and management area. Proper staffing and training are critical for success in initiating, implementing, and maintaining an enterprise architecture.

**References**

Although I have not cited any sources because this is excerpted from my own unpublished writings, the content of this chapter is rooted in the fundamentals expressed in the following books.
