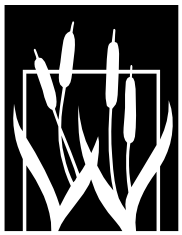

Whitemarsh
Information Systems Corporation

Enterprise Architectures

Whitemarsh Information Systems Corporation
2008 Althea Lane
Bowie, Maryland 20716
Tele: 301-249-1142
Email: Whitemarsh@wiscorp.com
Web: www.wiscorp.com

Table of Contents

1.0 Objective

This paper presents the Whitemarsh approach to the creation of architectures across the enterprise to ensure integration of missions, organizations, function, data, semantics, and policy. An ancillary objective is to show how these enterprise architectures are integrated with other classes of metadata within the Whitemarsh Knowledge Worker Framework and environment.

2.0 Topics Covered

The topics covered in this paper are:

- Enterprise Architecture Problem Space
- What are Enterprise Architectures?
- The Knowledge Worker Framework
- Five Knowledge Worker Framework Contained Architectures
- The Role of the Metabase
- Enterprise Architecture Way Ahead

3.0 Enterprise Architecture Problem Space

A lack of demonstrable integration across what could be argued to be various architecture classes are what prompted this paper. It is common for different teams to be assembled across the enterprise and to then "do their thing." One group does the enterprise's[1] architecture while another group does data models, and some third group does business information systems[2] plans.

The result is often three large piles of paper, and in the case of data models, really neat looking diagrams. Sometimes, Picasso comes to mind as the artist. Then everybody sits back and waits for the magic caterpillar to butterfly transformation to occur. Nothing seems to happen. A "winter of discontent" sets in. Everybody searches for the Spring, but it is nowhere to be found.

---

[1]. As defined within this book, an enterprise is merely a term to relate to a collection of organization units that have common collections of data, processes, activities within a business or a company and sometimes beyond corporate affiliations as in the case of data interchanges. An enterprise is therefore not just a synonym for business or a company. Rather, it is intended to convey a common data, process, and activity view across the organizational units sharing that view.

[2]. In this book, a business information system is a set of application-specific software that created, manipulates, evolves or deletes data–most commonly–from a database through a database management system in support of some mission area of the enterprise. A business information system is class of information system, and it is distinguished from other classes such as a computer's operating system, or "systems software" information system such as telecommunications management, database management systems, and end-user security management.

This paper shows that not only can Spring happen, but also that three architecture-based product sets, that is, the enterprise's architecture, the data models, and the information system plans are actually part of a larger collection of architectures that are tightly coupled, interdependent, integrated, and non redundant. In the process of creating the architecture product sets over the past 40+ years, its become painfully obvious that two architectures are missing, and that one architecture, data models, is badly engineered.

The two architectures that are missing are Database Object Classes and Resource Life Cycle Analysis. Database object classes are needed to overcome the real data and business perception problems that were caused bu the popularity of the relational data model.

The Resource Life Cycle Analysis is needed because of two reasons. First, Resource Life Cycle Analysis enables the creation of an integration lattice work between the enterprise's architecture products and the enterprise's databases and business information systems. The second reason is that not only does Resource Life Cycle Analysis makes business information systems planning possible in realistic time frames and with reasonable resources, it makes these plans able to be repeatable, reliable, and malleable.

The architecture that is badly engineered is data model. After extensive research into what data architects and data modelers actually do, a layered data architecture approach is necessary. This approach is not fundamentally new. Rather it sets down into actual practice what is actually done. This approach is more practical, is accomplished all the time by data architects and data models but without persistence and repeatability. The result is a collection of data architecture products that support maximal reuse, complete integration, and very controlled redundancy.

In addition, the data architecture approach completely integrates the only two ISO/ANSI standards in this area: ISO 11179 for data element metadata, and ISO/ANSI SQL for comprehensive, portable, and interoperable data model development.

This paper thus sets out five distinct data architectures. While that's good, what integrates the work and the work products from all these architectures? It is the Knowledge Worker Framework. This framework, first explicitly posited in 1992, was squarely founded on a database project methodology that has been successfully evolving over the past 30 years. Further, the organization, sequencing, and work products in the Knowledge Worker Framework are those that satisfy 100% of the U.S. General Accountability Office's reasons for information technology system failure. These reasons were discovered from analysis of 13 multi-hundred million dollar information technology failures.

It is not enough however, to just assert integration. It must be proven. Thus, there is an underlying metadata repository system that supports the creation, manipulation, evolution, and integration of all the work products. It is upon this highly engineered, experience-fired foundation that this paper is based.

4.0 What are Enterprise Architecture?

The term, Enterprise Architecture contains both Enterprise and Architecture. In Wikipedia, enterprise contains no special definition. Rather, the Wikipedia states that "almost any business or organization can be called an enterprise." From the Whitemarsh book, Data Interoperability Community of Interest Handbook, an enterprise is:

> *... a term to relate to a collection of organization units that have common collections of data, processes, activities within a business or a company and sometimes beyond corporate affiliations as in the case of data interchanges. Ideally, there is also a common governance of these items. An enterprise is therefore not just a synonym for business or a company. Rather, it is intended to convey a common data, process, and activity view across the organizational units sharing that view.*

As to architecture, the Wikipedia states that an architecture is:

> *... the design of the total built environment, from the macrolevel of town planning, urban design, and landscape architecture to the microlevel of creating furniture. Architectural design usually must address both feasibility and cost for the builder, as well as function and aesthetics for the user.*
>
> *Planned architecture often manipulates space, volume, texture, light, shadow, or abstract elements in order to achieve pleasing aesthetics. This distinguishes it from applied science or engineering, which usually concentrate more on the functional and feasibility aspects of the design of constructions or structures*

The architectures contained in this paper are in the class: Science and Engineering. Specifically, the Wikipedia provides a definition for Enterprise Architecture as follows:

> *Enterprise architecture is the practice of applying a comprehensive and rigorous method for describing a current and/or future structure and behavior for an organization's processes, information systems, personnel and organizational sub-units, so that they align with the organization's core goals and strategic direction. Although often associated strictly with information technology, it relates more broadly to the practice of business optimization in that it addresses business architecture, performance management, organizational structure and process architecture as well.*

5.0 Knowledge Worker Framework

A knowledge worker is someone who primarily works with information and abstract concepts. Another type of worker is the real product worker. White collar workers such as clinicians and clinical support personnel are knowledge workers because they develop care plans, provide treatments, and record results. Alternatively, workers on a manufacturing line and for example, food service personnel are not knowledge workers because they are primarily focused on the creation and/or assembly of real products.

Both knowledge workers and real product workers share common characteristics including plans, schedules, estimates and result assessments. Notwithstanding, the fundamental work methods and environment that underlie the knowledge worker and the real product worker are different at the core. Thus, trying to make one a clone of the other is both frustrating and invalid.

Due to the abstract nature of their work, information required by knowledge workers can best be stored, assimilated and used as objects, which are encapsulations of data and processes. To most effectively support knowledge workers, the enterprise should strive to create object-oriented environments.

These two concepts, knowledge worker and object-oriented environments are brought together into technology architectures since both uniquely characterize the ideal working environment.

The knowledge worker's environment involves both automated and non automated activities. Some non automated activities involve the use of automation, for example, once a patient receives a treatment from a clinician (non automated activity), the characteristics of the treatment, and the clinicians observations about the patient's reaction to the treatment are typically recorded in some automated system. A knowledge worker's framework must therefore address manual and automated activities.

Knowledge workers perform groups of functions to accomplish their designated job or to accomplish some aspect of the enterprise's mission. Knowledge workers may perform these function groups in different combinations depending on the enterprise's organization. For example, if an organization is highly distributed into multi-functional units, there may be staffs that perform diverse groups of functions. Conversely, a highly centralized organization may have certain staff devoted to specific and highly specialized functions. The knowledge worker is therefore a complex multi-faceted person who performs diverse functions of different complexities for one or more organizations.

Knowledge workers need a framework within which all their work products are created, stored, and interrelated. All the architectures are thus set within the Knowledge Worker Framework.

The Knowledge Worker Framework's work products are completely defined, integrated and non redundant. They are supported by an in depth methodology, books, other papers, workshops, seminars, and of course the metabase software system that stores, interrelates,

updates, and reports all Knowledge Worker Framework products through a multi-user SQL database and system. Table 1 sets out the Knowledge Worker Framework.

Table 2 provides a summary of the percents of the reasons for failure that the Knowledge Worker Framework's work products address. A most interesting row in Table 2 is the last one. It shows that only 5% of all of the U.S. GAO reasons for failure can be attributed to information technology.

If all the cells' work products are accomplished appropriately, there's success is almost certain. But if collections of cells are not done or done inappropriately, there is a moderate to sever probability of failure. Table 2 again provides the key reasons for failure percents and summary descriptions of the reasons.

Tables 3 and 4 provide a high level description of the overall intent of the rows and columns of the Knowledge Worker Framework. Each row and/or column lays out the set of knowledge worker products that have to be created to achieve the overall intent of the row or column. Each row or column does not exist in isolation. There are relationships within and between the cells and columns. Some of the relationships are between non adjacent cells. What integrates it all are a detailed methodology and the metabase system which captures, stores, interrelates, and reports all the knowledge worker's products. Detailed descriptions can be found in materials cited in the References section of this paper.

Enterprises commonly create computing supports for knowledge workers under the assumption that the functions they perform and the organizations through which they act are fixed and seldom change. Not only are these assumptions wrong, but when the functions and organizations do change, computing environment changes seldom keeps pace because they are time consuming to specify, difficult to implement, and slow to accomplish. Slow-to-react computing environment changes, therefore, become the very reason why information technology support to business functions and organizations cannot keep pace with the demands of change. What is needed are computing environments that are object-oriented, sensitive to knowledge worker functions and organizations, and that can react to the demands of change in a timely fashion.

Key to all these architectures is the organizing framework for the knowledge worker, that is, the knowledge worker's framework. This framework organizes and interrelates these five interconnected and contained architectures and enables all the architecture work products to be integrated, and non redundantly stored in one metadata repository. This metadata database repository, hereinafter called the metabase, supports the creation, manipulation, and reporting of only what the enterprise needs to exist and thrive.

The material about the Knowledge Worker Framework contains, at varying levels of detail, descriptions about all the products, including the products contained in the architectures encompassed by the Knowledge Worker Framework. The sections on the individual five architectures, for example, Database Object Classes, provide additional levels of detail about Database Object Class components, reasons for the need for Database Object Classes, the relationship among the components of Database Object Classes, and the relationship between components Database Object Class and components from other architectures.

Knowledge Worker Framework						
		Man-Machine Interface				
		Machine		Interface	Man	
Deliv-erables	Mission	Database Object Class	Business Information System	Business Event	Business Function	Organization
Scope	List of business missions	List of major business resources	List of business information Systems	List of interface events	List of major business scenarios	List of organizations
Business	Mission hierarchies	Resource life cycles, Data Elements, Specified Data Model	Information sequencing and hierarchies	Event sequencing and hierarchies	Business scenario sequencing and hierarchies	Organization charts, jobs and descrip-tions
System	Policy hierarchies	Database Object Class models	Information system designs	Invocation protocols, input and output data, and messages	Best practices, quality measures and accomplishm ent assessments	Job roles, responsi-bilities, and activity schedules
Techno-logy	Policy execution enforcement	Implemented Data Model	Information systems application designs	Presentation layer information system instigators	Activity sequences to accomplish business scenarios	Procedure manuals, task lists, quality measures and assessments
Deploy-ment	Installed business policy and procedures	Operational Data Model	Implemented information systems	Client & server windows and/or batch execution mechanisms	Office policies and procedures to accomplish activities	Daily schedules, shift and personnel assignments
Opera-tions	Operating business	View Data Models	Operating information systems	Start, stop, and messages	Detailed procedure based instructions	Daily activity executions, and assessments

Table 1. Knowledge Worker Framework.

U.S. General Accountability Office Reasons for Information Technology System Failure	
Percent	**Reasons for information technology Failures Description**
41%	A lack of proper identification, analysis and configuration of enterprise architecture Scope and Business Rows of the Missions, Database Objects, Business Information Systems, Business Events, Business Functions and Business Organizations.
29%	A lack of proper identification, analysis and configuration of enterprise architecture Scope and Business Rows of just the Missions, Business Functions and Business Organizations.
50%	A lack of proper re-engineering of the business functions and organizations as a consequence of re-engineered databases and business information systems.
5%	A lack of proper engineering and development of databases and business information systems.

Table 2. Summary of U.S. General Accountability Office Reasons for Information Technology System Failure

Knowledge Worker Framework Rows	
Scope	The Scope row discovers, enumerates, interrelates, and, at a high level the enterprise's missions, Database Object Classes, business information systems, business event interfaces, business functions, and organization.
Business	The Business row details the objects that have been discovered and presented in a high-level way from the scope row. In addition to further detailing, the missions, organization, and functions are all set within hierarchies.
Systems	The Systems row is the first row devoted to a "systems perspective." It presumes that the artifacts created for this row will be employed during the creation of a system. Not all systems are to be information technology systems, however. Some systems are just a systematic set of policies that guide the accomplishment of a highly engineered set of human activities, that is, functions.
Technology	The Technology row, is similar to the Systems row in that it represents a detailing and a new set of artifacts needed to support either policy specification, Business Organization and Business Function specification, or a furthering of the efforts in support of IT.
Deployment	The Deployment row presumes that the new environments, from Mission through Business Organizations are ready to be deployed prior to operations. Essentially this overall row is a roll-out of all the artifacts so that they can be employed.
Operations	The Operations row represents the new operating environment. The mission cell would be the operating business. Feedback mechanisms would occur from every set of organization-based functions. This will then enable a feed back cycle at least one row above. There would be a similar set of operations for the Business Functions and the Business Organizations.

Table 3. Knowledge Worker Framework Rows and Descriptions.

Knowledge Worker Framework Columns	
Mission	The Mission column represents the rationale or basis for the knowledge worker environment. Missions include policies, policy execution enforcement, installed policies and procedures, and then the operations of those policies including feedback.
Database Object	A Database Object Class column is a collection of traditional (that is, formatted and structured data) and nontraditional (that is, video, sound, and unstructured text) data. Database object classes proceed through precisely defined states starting with the null state, and then a series of discrete business defined, interlinked non-null states, and finally a null state.
Business Information Systems	The Business Information System column represents computer-based data processing systems that accomplish database object state transformations from within the context of business functions. Different business functions may cause the execution of the same business information system. If, in any of the business functions that employ a business information system, the database object value state transformation is not accomplished, the entire set of database object transformations is rolled back so that the database object returns to its prior state.
Business Event	The Business Event column represents the interfaces between the two "machine" columns of the framework and the two "man" columns. The main reason there is a column formally dedicated to the interfaces between man and machine is to preserve their independence, and to set these events squarely within calendar and business cycles.
Business Function	The Business Function column represents the procedures accomplished by someone within an organization to complete some aspect of a business' mission. Business functions almost always exceed the bounds of business information systems.
Business Organization	The Business Organization column represent formally constituted groups of persons chartered to perform business functions to achieve some aspect of a business' mission. While small businesses often have the same organization from one location to another, large businesses do not. In fact, as businesses become larger and more diversified, organizations become different, stylized, and whenever management changes, business organizations often change in lock-step.

Table 4. Knowledge Worker Framework Columns and Descriptions

6.0 Five Knowledge Worker Framework Contained Architectures

The definition set out in the previous section sets out the context for the five enterprise-related architectures which are identified in Table 5. The five architectures presented are all contained within the framework for the knowledge worker. All together, these individual architectures enable the enterprise to engineer, evolve, and know answers to the following questions:

- What is the enterprise's mission, organization, functions and domains of data?

- What is the key policy-based engineered data that has to be coherent, integrated, and non redundant?

- What the enterprise resources are and how they progress individually and in concert to achieve enterprise missions in the most efficient and effective manner possible?

- What is the overall business information systems plan must be and in what sequence database and business information system projects must be accomplished to achieve just-in-time readiness?

Enterprise Architectures	
Architecture Class	**Brief Description**
Knowledge Worker Framework	An architectural framework from Wikipedia is a "...skeleton upon which various objects are integrated for a given solution." In this case, the scope of the integrated solution is the "knowledge worker."
Enterprise	In the architecture for the enterprise itself, the engineering and structure of the enterprise's mission, organizations, functions and database domains are provided such that they can be extended and/or integrated with other more technical architectures such as hardware, information systems, and business events.
Database object classes	A Database Object Class architecture is a self-contained set of data structures, processes, states and state-transformation systems that advance a collection of highly engineered data within the domain of a database management system (DBMS).
Data	A data architecture is the set of data-specific structures that enable enterprise-wide data and semantics integration critical to data interoperability.
Resource Life Cycle Analysis	A Resource Life Cycle Analysis is a set of enterprise resources, each set within a life cycle of major state transformations of those resources. These Resource Life Cycles can be interconnected, and then be used as a lattice work to attach the enterprise's inventory of database and information systems, which in turn, greatly assists in the formation of business information systems plans.
Business information systems plans	A business information systems plan is a highly engineered sequence of the development of databases and information systems throughout the enterprise so as to cause maximum reuse and to accomplish the building and evolution of databases and information systems in a business-rationale-based and highly ordered fashion.

Table 5. Examples of Enterprise Architectures in a Knowledge Worker environment.

Table 6 presents where these five architectures fit within the Knowledge Worker Framework that is presented in Table 1.

The architecture is listed in the first column of Table 5. The second and third columns provide the "coordinates" of the contained architectures in terms of Knowledge Worker Framework rows and columns.

Knowledge Worker Framework Contained Architectures		
Architecture	**Rows**	**Columns**
Enterprise	Scope and Business	Mission through Organizations
Database object classes	System and Technology	Database object class
Data	Business, System, Technology, Deployment, and Operations	Database object class
Resource Life Cycle Analysis	Business	Database object class
Business information systems plans	Scope through Technology	Mission, Database object class, and Business Information Systems

Table 6. Knowledge Worker Framework contained architectures.

The tabulated effects the five architectures make to the GAO failure reasons are provided in Table 7. The Knowledge Worker Framework columns are presented across the top, and the Knowledge Worker Framework rows are presented along the side.

Each of the Table 7 cells identifies the general scope and purpose of the contained work products. If one of the five architectures is also contained within that cell then it is listed along with its approximate percent contribution to the elimination of the GAO reasons for failure. You will quickly note that the sum of the percents exceeds 100. That is because the allocation in this type of table has to be general.

Table 7 clearly shows that the Enterprise's Architecture is the most critical as it addresses the highest percent of failure reasons. The Function and Organization cell (lower right) changes are outside of the five defined architectures. This shows that all cells and work products from the Knowledge Worker Framework are important to address even if they are outside any given architecture. It finally shows that the Information Systems Planning, Resource Life Cycle, and Data Architectures all address a small percent of the overall failure reasons. These three lower-percent architectures need to be done however, not just because they contribute to correctness, but also because if done well they significantly contribute the overall efficiency of all the work products of the knowledge worker.

<table>
<tr><th colspan="7">Knowledge Worker Framework</th></tr>
<tr><th></th><th>Mission</th><th>Database object class</th><th>Business Informa-tion system</th><th>Business Event</th><th>Function</th><th>Organ-ization</th></tr>
<tr><td>Scope</td><td colspan="6" rowspan="2">Enterprise Architecture (41% of failure reasons)</td></tr>
<tr><td>Business</td></tr>
<tr><td>Business information system</td><td colspan="3">Business Information Systems Planning Architecture (8 % of failure reasons)
Resource Life Cycle Analysis Architecture (< 5% of failure reasons)
Data Architectures (< 5% of failure reasons)</td><td colspan="2" rowspan="4">Function and organization changes to the enterprise to take advantage (50% of failure reasons)</td></tr>
<tr><td>Technology</td><td rowspan="3">Mission changes post imple-mentation</td><td colspan="2" rowspan="3">IT database and business information systems creation, deployment, and maintenance
Data Architectures (<5% of failure reasons)</td></tr>
<tr><td>Deployment</td></tr>
<tr><td>Operations</td></tr>
</table>

Table 7. Distribution of metadata across the knowledge worker framework.

6.1 The Enterprise's Architecture

The first architecture, the enterprise's architecture, provides the overall context within which the knowledge worker operates. The enterprise's architecture identifies and describes the enterprise's mission, organization, functions and database domains. It also defines of data which ultimately provide the "proof of execution" of the operating enterprise.

The data architecture and resource life cycle architecture are dependent architectures. They are dependent because their scope's depend on the content of the enterprise's architecture. All the data specifications, resource specifications, and resource life cycles are all bound by the scope of the enterprise's mission.

It is therefore appropriate that the enterprise's architecture is accomplished first as it is from within this architecture that all other architectures must fit.

However, if other architectures are being done in parallel, then that's acceptable just so long as the work products that overlap between the architectures are reconciled. Ideally while there may be overlapping work products there will only be one set. For example, if there's a resource and one or more of its life cycle nodes cannot be discovered from within the enterprise's missions, then an analysis error has been made. Such errors must be resolved.

6.2 Database Object Class Architecture

The second architecture, Database Object Classes represent the objects that are valued, updated, employed, and deleted by the knowledge worker as the enterprise's business is conducted. Proof of mission accomplishment exists in the form of data. The data must, however, be consistent, highly engineered, integrated, and non redundant. Database object classes provide such constructions.

The unique role Database Object Classes play in a database environment is that the Database Object Class is completely encapsulated within the DBMS schema definition layer and therefore cannot be avoided by any business information system regardless of source language form.

There is a real business case for database objects. Database object classes can be implemented in SQL DBMSs that are not object-oriented as well as SQL:1999 or more advanced DBMSs that are object-oriented.

All the different Database Object Class work products were described and were interrelated. Shown also was how these work products fit with the work products of the other architectures.

6.3 Data Architectures

There are two dimensions to data architecture: Database architecture classes, and data model generalization levels. Database architecture classes are a way of characterizing every database that exists in the enterprise. Each class has peculiar characteristics and is designed to meet certain data and integrity objectives.

Data model generalization levels are a way of managing all specified data in the enterprise regardless of its database architecture class. Two of the data model generalization levels, i.e., data elements and specified data models are commonly shared across database architecture classes.

6.3.1 Database Architectures

The database architecture classes, depicted in Figure1 are:

- Original data capture.
- Transaction data staging area.
- Subject area databases.
- Data warehouses (wholesale and retail).
- Reference data.

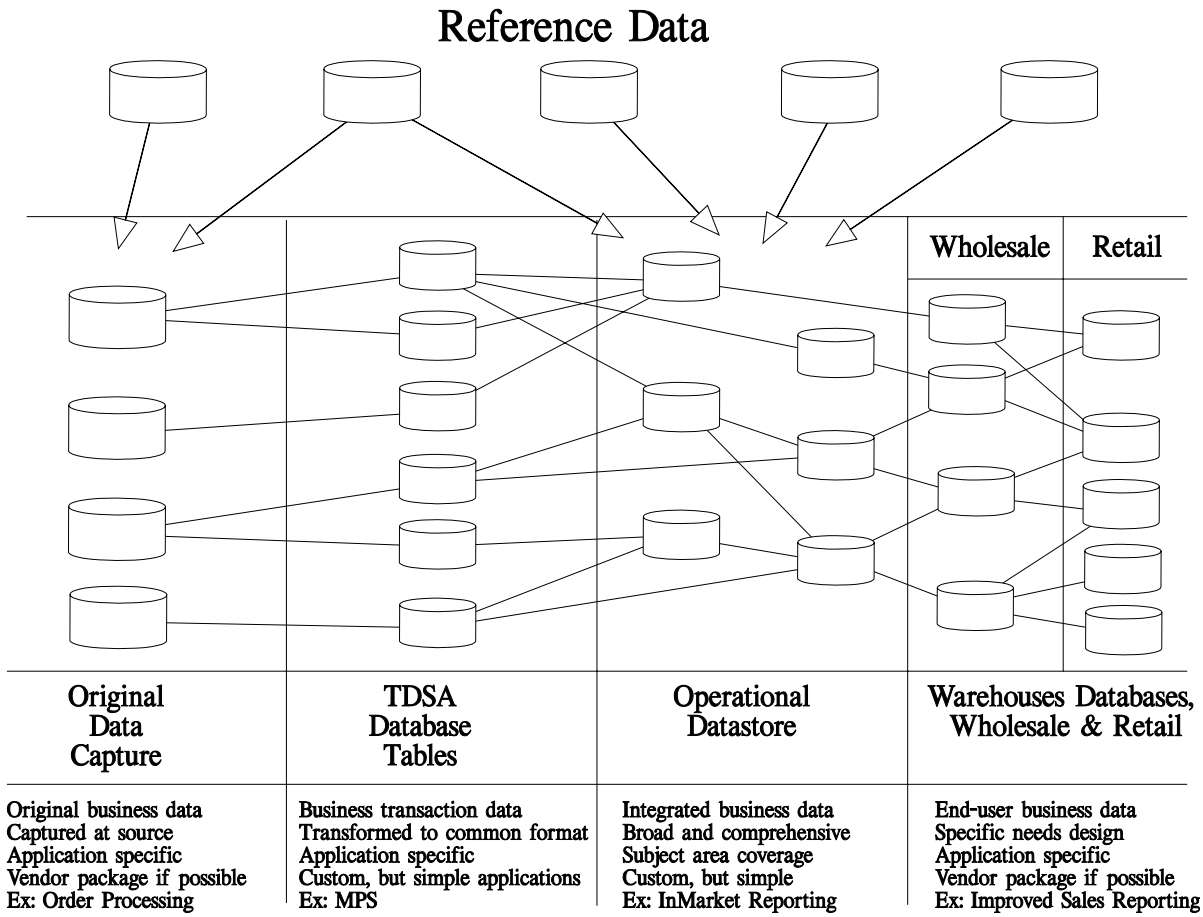

Figure 1. Database Architecture Classes.

The original data capture database architecture class represents the actual databases that reside within organizations within an enterprise and may provide data to other database classes. These database types are often called OLTP databases because they support on-line transaction processing.

The transaction data staging area database architecture class represents the data extracts from the original data capture databases that are then, if necessary, modified to the required semantics for any of the other database architecture classes. Any interface between any database architecture class may proceed through a transaction data staging area.

The subject area database architecture class represents the subject-based integrated databases of data that, in turn, support some measure of analyses and reports, analysis results retention, and supports the generation of other classes of databases, that is, data warehouses.

The data warehouse (wholesale and retail) database architecture represents the transformed and likely redundant sets of data that serve special reports and analyses. The key set

of differences between wholesale data warehouses and retail data warehouses are one of volume, duration and specialization. Data mart data warehouse designs are commonly created along the lines of "star schemas" or "snow-flake schemas," and when compared to wholesale a data warehouse have smaller volumes, shorter durations, and are more specialized.

The reference data database architecture class represents data that form the critical characterization and discrimination characteristics of data from within the entire set of business facts. Included, for example are genders, city names, state names, all codes, and like.

Ideally, all reference data would be exactly the same across all the other four data architecture classes. Realistically, however, different agencies and providers have different reference data value versions for the same reference data and different data value versions across time. All reference data must be managed centrally and then distributed in so far as it is possible to all databases of the other data architecture classes.

6.3.2 Data Model Generalization Levels

There are five data model generalization levels that range from enterprise-wide data elements and data models of shared function-based concepts to technology independent database designs, SQL DBMS dependent designs, and finally, interface specifications between databases controlled by a DBMS and business information systems.

Every database architecture class employs and/or proceeds through the following data model generalization levels:

- [11179] Data elements.
- Specified data models.
- Implemented data models.
- Operational data models.
- View data models.

Database object classes are defined within the Implemented data model and made operational during the Operational data model. These five levels, described in Table 8, and are illustrated in Figure 2.

Data Model Generalization Classes	
Name	**Description**
Enterprise Data Elements	Data elements are enterprise-level business fact definitions independent of use within specific databases. Data elements are defined according to the requirements of ISO 11179 for data element metadata. Additionally, data elements include semantic and data use modifiers so as to enable automatic name construction, automatic definitions, and automatic abbreviations. There is a one-to-many relationship between data elements and Specified

Data Model Generalization Classes	
Name	**Description**
	Data Models.
Specified Data Models	Specified Data Models are data models of concepts that are independent of use within specific databases. These Specified Data Models of concepts are defined as entities, attributes and relationships within specific subjects. Entities can be related across subjects. There is a many-to-many relationship between the Specified Data Model and the Implemented Data Model.
Implemented Data Models	Implemented data models are models of databases that are independent of any particular DBMS such as Oracle or DB2. The Implemented Data Models obtain their data structures from one or more Specified Data Models, and in turn provide these database data structures for use in the construction of data models of databases that actually operate under specific DBMS, and run on computers. These Implemented Data Models are defined as tables, columns and relationships within schemas. Tables cannot be related across schemas. All the tables in this data model generalization level are in at least third normal form so as to ensure data model clarity and quality design. There is a many-to-many relationship between an Implemented Data Model and an Operational Data Model.
Operational Data Models	Operational data models are models of databases that conform to the requirements of a particular DBMS such as Oracle or DB2. Additionally, the database's design must conform to the expected processing requirements of the particular set of database applications supported by the particular database. Operational Data Model designs may vary because of many factors such as transaction volume, the host operating system, and the computer hardware size and throughput capabilities. Regardless, every variation of a given Operational Data Model database is related back to the Implemented Data Model of the database. These Operational Data Models are defined as DBMS tables, DBMS columns and relationships within DBMS schemas. "DBMS" is employed here to signify that these data model components are tied to a specific DBMS. There is a one-to-many relationship between an Operational Data Model and a View Data Model.
View Data Models	View data models are the business information system and Operational Data Model intersection mechanisms. View data models consist of views and view columns. They are also related to the business information systems that are the sources or targets of the database data. Views are specifically tied to specific DBMSs and view columns from a source view may be mapped to the view columns of a target view. Finally, view columns may be computed or derived.

Table 8. Data model generalization levels.

From Figure 2, the highest level of generalization is ISO 11179 data elements. This level enables the singular definition of the maximally used business facts and the setting of these business facts within business concepts and controlled value domains.

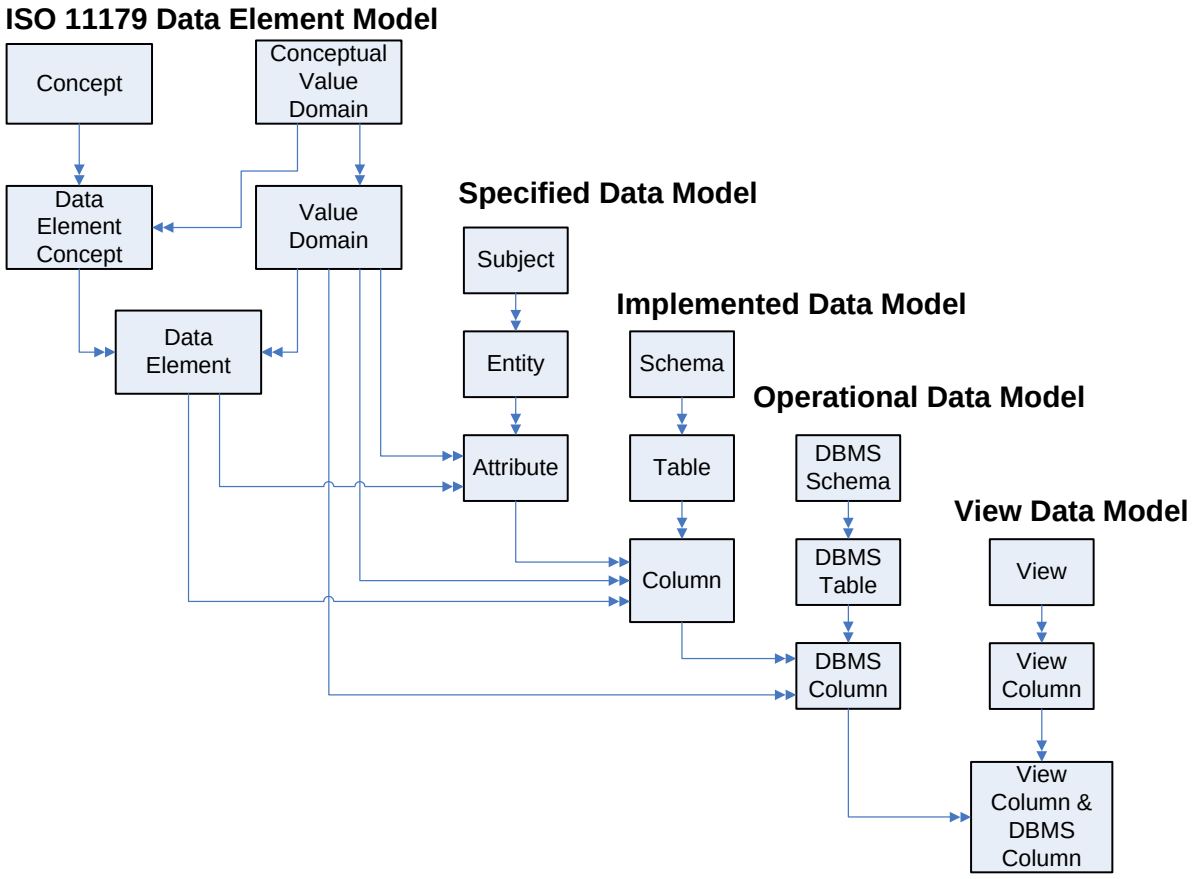

Figure 2. Data Model Generalization Levels.

The second level of generalization, Specified Data Models, enables the specification of standardized functionally-based data models of concepts that ultimately, as whole or partial collections get used over and over during the construction of implementable database models.

The third level of generalization, Implemented Data Models, represents data models of databases that may ultimately become one or more variations of the operating databases. Additionally, whole or partial data structure collections from these Implemented Data Models may be employed in one or more of the operating databases.

The fourth level, Operational Data Models, represent the operating databases that reside on computers and ultimately provide the data to business information systems.

The fifth level, View Data Models, represent the level of data exchange between database and application information systems. The exchange level has two forms, technology bound, and technology independent. The technology bound data exchange level, SQL views, support the

direction interaction between the database management system (DBMS) databases and the business information systems.

The technology independent exchange level is commonly seen in the form of XML, which has two parts: definition, and data stream. Combined and then either composed by a business information system, or shredded into a database's required structure by the DBMS, data is able to be put and retrieved by DBMS-disconnected business information systems. XML is not however a know-nothing, do-everything data exchange environment. The four layers above the XML layer (Operational Data Models through data elements) must already exist for XML to be effective.

The point to these five generalization levels is that they:

- Enable order out of data specification chaos,
- Integration and reuse of semantics from the current discord,
- Control of or even elimination of large scale redundancy of data across the enterprise.

Ultimately, database should promote harmony, civilized discourse, and not be the cause of the bureaucratic food-fights. These data model generalization levels contribute significantly to achieving this objective.

6.4 Resource Life Cycle Analysis Architectures

The fourth architecture, resources and their life cycles are an enterprise resource organizing architecture. The key premise is that every enterprise consists of a finite set of well-known resources that are created, manipulated, and ultimately dissolved by the enterprise. Examples include locations, facilities, products, monies, staff, real and abstracts, and the like. Enterprise operations are thus seen through the operating life cycles of these resources.

These Resource Life Cycles are also interconnected, one life cycle with another. Collectively, the interconnected Resource Life Cycles form a lattice work through which missions, organizations and functions are operationally viewed, and upon which database objects and business information systems are integrated and coalesced.

Resource Life Cycle Analysis enables business mangers to participate in the identification of the resources, their life cycles, and then enablement vectors among the life cycles.

The information technology staff then has a business management defined network against which it can allocate the existing information systems and databases. Once allocated, information technology can root out the conflicting and redundant systems and databases.

Figure 3 illustrates a resource life cycle network among three different resource life cycles. The three life cycles are court personnel, case, and document. The story that this diagram represents is as follows: Court personnel must be assigned before a case can be classified. Then, once a case is processed, a particular document can be acted upon.

When these interrelationships are brought into the development or evolution of collections of business information systems and databases, those associated with the first three nodes of a Court personnel Resource Life Cycle must be in place and operational before any court personnel can be Assigned. Similarly, the databases and business information systems associated with the File node of the Case Resource Life Cycle must be in place before a Case can be classified. The databases and business information systems up to and including the Process Resource Life Cycle node of the Case Resource Life Cycle must be in place before any document can be Acted Upon. Finally, the database and business information systems up to and including the Act Upon Resource Life Cycle node of the Document Resource Life Cycle must be in place before a document can be acted upon.

Figure 4 illustrates the allocation of a collection of databases and business information systems onto a resource life cycle. The resource life cycle network is thus an excellent way of allocating the inventory of all databases and business information systems. Immediately the metabase environment which then stores this inventory can report on candidate redundancy and obvious gaps.

Information technology projects, representing advances in the sophistication of its support to the business, can be identified and resource-estimated through standardized methodologies and metrics.

Once a set of information technology projects is estimated, the entire set enterprise-network of information technology projects can be scheduled through project management software. At that point, management re-enters the picture and makes the hard choices, what, when, and why. Information technology is left with the How. That's as it should be.

It is again important to note that while valuable in their own right, Resource Life Cycles also serve as a key linking mechanism between enterprise, database objects, and data architectures on the one hand and business information systems plans on the other hand.

6.5 Business Information Systems Plan Architectures

The fifth architecture, business information systems planning, is mainly an architecture that reflects the build and evolution sequence of all information technology projects. This business information systems planning architecture is built directly from the foundation of the Resource Life Cycle Analysis architecture, which in turn was built from the database objects, data, and enterprise architectures.

Business information system plan architectures are both blueprints of all information technology database and business information system projects and also the battle plan for their accomplishment. Given that the percent of effort devoted to analysis and design of any individual business information systems project is about 20% of the first implementation cycle, allocating just 3% to the development of the enterprise's business information systems plan is not out of line.

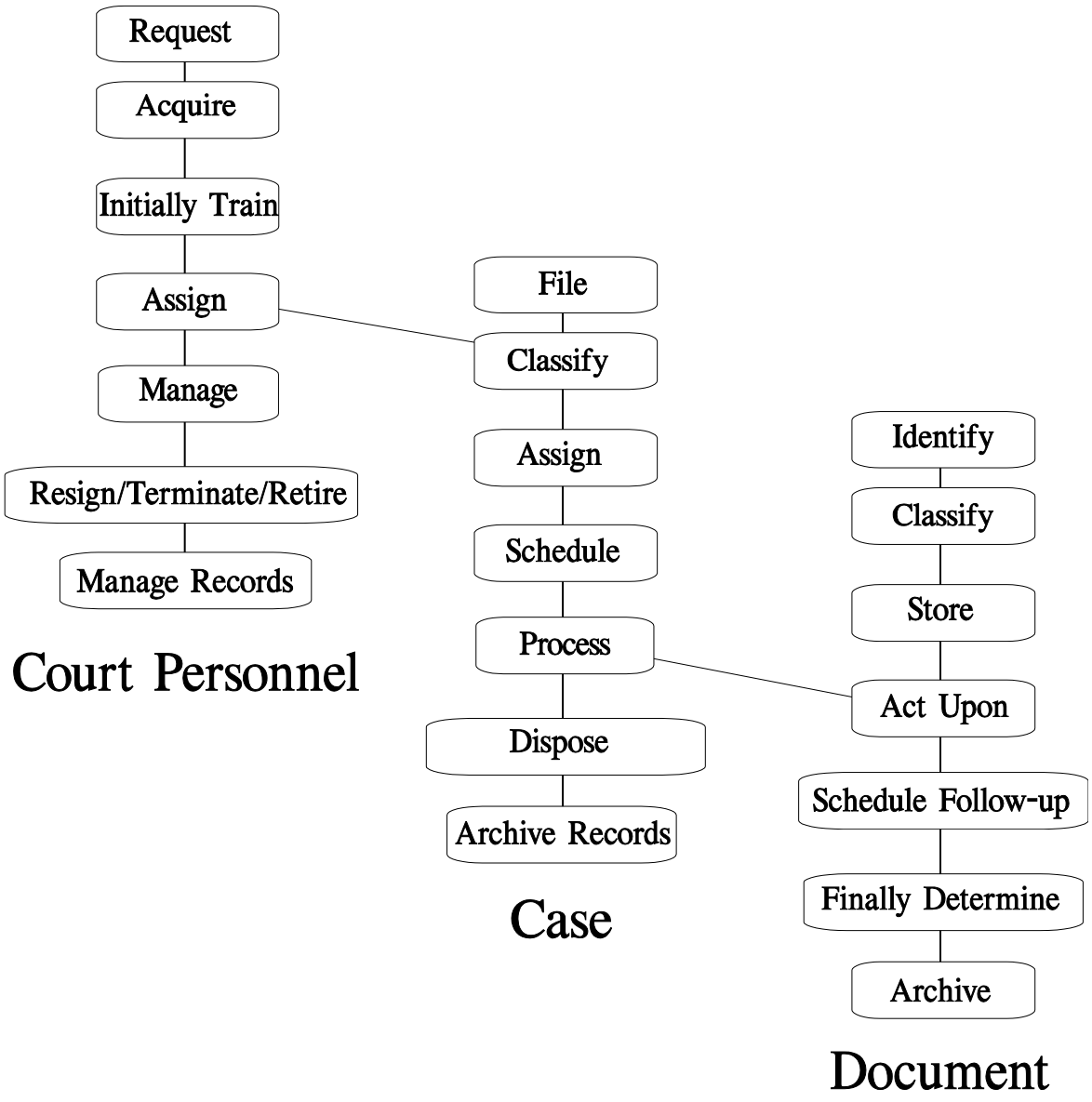

Figure 3. Resource Life Cycle Network.

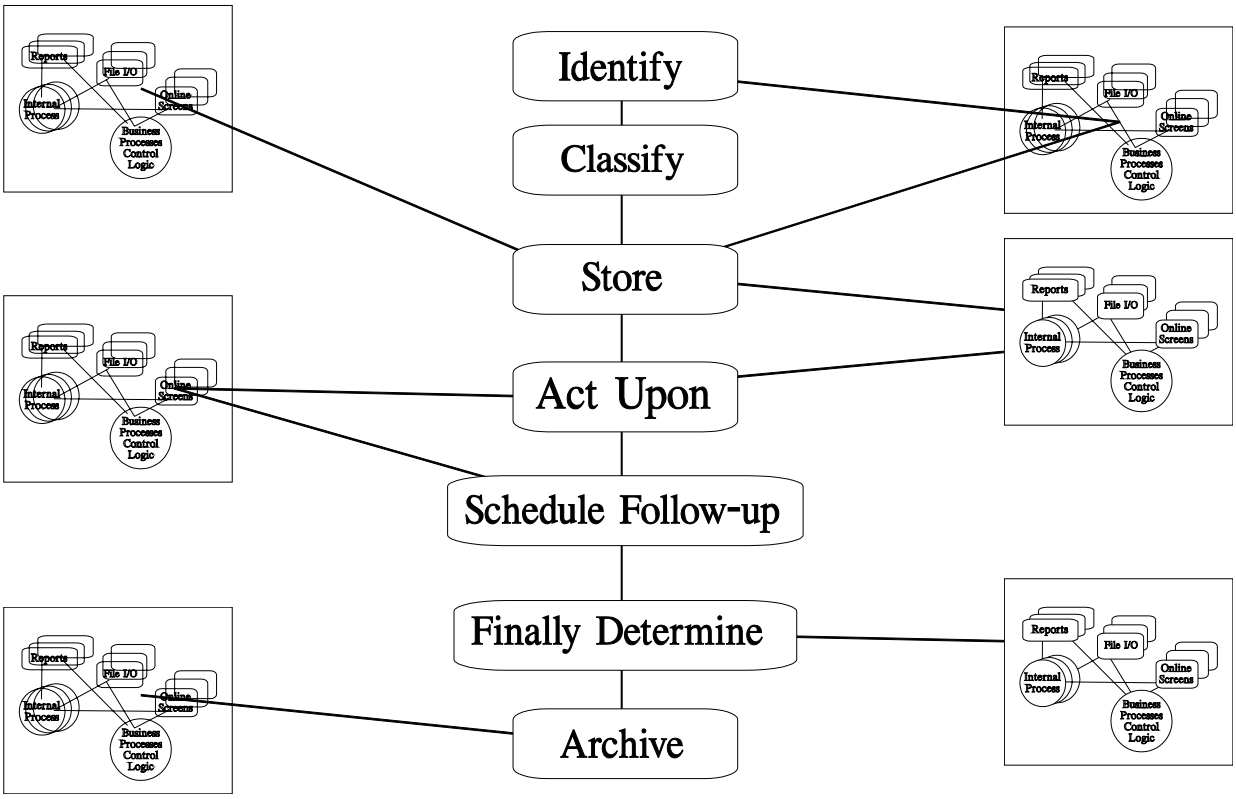

Figure 4. Allocation of Databases and Business Information System to a Resource Life Cycle.

6.5 Business Information Systems Plan Architectures

The fifth architecture, business information systems planning, is mainly an architecture that reflects the build and evolution sequence of all information technology projects. This business information systems planning architecture is built directly from the foundation of the Resource Life Cycle Analysis architecture, which in turn was built from the database objects, data, and enterprise architectures.

Business information system plan architectures are both blueprints of all information technology database and business information system projects and also the battle plan for their accomplishment. Given that the percent of effort devoted to analysis and design of any specifice business information systems project is about 20% of the first implementation cycle, allocating just 3% to the development of the enterprise's business information systems plan is not out of line.

A quality business information systems plan must, of course, be timely, useable, maintainable, able to be iterated into a quality product, and reproducible. A description of these characteristics is provided in Table 9.

Business information systems plan Components	
Characteristic	**Description**
Maintainable	The business information systems plan must be maintainable. New business opportunities, new computers, business mergers, etc. all affect the Business Information Systems Plan. The business information systems plan must support quick changes to the estimates, technologies employed, and possibly even to the fundamental project sequences. Once these changes are accomplished, the new business information systems plan should be just a few computer program executions at most a few days away.
Quality	While the business information systems plan must be a quality product, no business information systems plan is ever perfect on the first try. As the business information systems plan is executed, the metrics employed to derive the individual project estimates become refined as a consequence of new hardware technologies, code generators, techniques, or faster working staff. As these changes occur, their effects should be installable into the data that supports business information systems plan computation. In short, the business information systems plan is a living document. It should be updated with every technology event, and certainly no less often than quarterly.
Reproducible	The business information systems plan must be reproducible. That is, when its development activities are performed by any other staff, the business information systems plan produced should essentially be the same. The business information systems plan should not significantly vary by staff assigned.
Timely	The business information systems plan must be timely. A business information systems plan that is created long after it is needed is useless. In almost all cases, it makes no sense to take longer to plan work than to perform the work planned.

Business information systems plan Components	
Characteristic	**Description**
Useable	The business information systems plan must be useable. It must be so for all the projects as well as for each project. The business information systems plan should exist in sections that once adopted can be parceled out to project managers and immediately started.

Table 9. Characteristics of a quality of a business information systems plan.

For this Information Systems Plan strategy to be successful, it must be supported by a database project methodology, and an already existing set of architectures for the enterprise, data, Database Object Classes, and resource life cycles. The reason these architectures should already exist by the time the business information system planning effort begins is because if done, the plan will be seen as a capstone and integrator of the other architectures, and the plan will be much more readily accepted because the other architectures, accomplished by collaborative enterprise organizations will represent a consensus-based foundation.

In contrast, if the business information systems plan is accomplished in isolation and within it scope is the creation of the enterprise, data, Database Object Classes, and resource life cycles architectures then there will be two unrecoverable problems. The first is that the plan development time will greatly exceed the 3% of the information technology budget, and second, these encapsulated architectures will have a much lower probability of being of value to the enterprise as a whole. There will have been no enterprise-wide sweat-equity an ownership in the architecture products. In short, business information systems plans done this way take longer, cost more, and most often just become “shelf-ware.” This is not good.

Unfortunately, these characteristics have been glaringly unfilled by other business information systems plan development methodologies. Notwithstanding the bad reputation of business information systems plan efforts, they still need to be done. This paper highlights that a better way exists and in the end will make business information systems plan part of the standard planning steps of a well run information technology organization.

Finally, the very process of configuring a final business information systems plan involves intense database and business information system project planning. This process too can be largely automated through the use of sophisticated project planning software where the PERT chart is the Resource Life Cycle Node network, and the individual projects associated with the nodes are easily estimated because of a large quantity of project plan and metric templates.

If project plan and metric templates are subsequently refined over time, then the ability to automatically generate new business information systems plans will become quite easy. Further, these generated plans will be reliable, repeatable, and are malleable as new technologies unfold.

7.0 Role of the Metabase

The metabase is just a database encapsulated in a metabase oriented business information system that has as its sole purpose the capture, manipulation, and reporting of metadata. The objectives of such a metabase system are these:

- Capture requirements and design artifacts.
- Support the extraction of design artifacts to then feed code generators and schemas for database management systems.
- Capture of enterprise operational metadata

The first objective is the most important because it addresses the highest percent of GAO reasons for failure. It is important to note, one more time, that most of the GAO reasons were outside the proper boundaries of information technology.

The second objective is largely accomplished by code generators, which, as a byproduct, automatically create throw-away metadata. The overall process of integration of the metabase with a code generator is depicted in Figure 5. The metadata is captured and stored on the left panel of this figure. A set of metadata that represents the data model is exported and then imported into either the code generator system, for example, Clarion, or into an SQL DBMS to then create the database's schema.

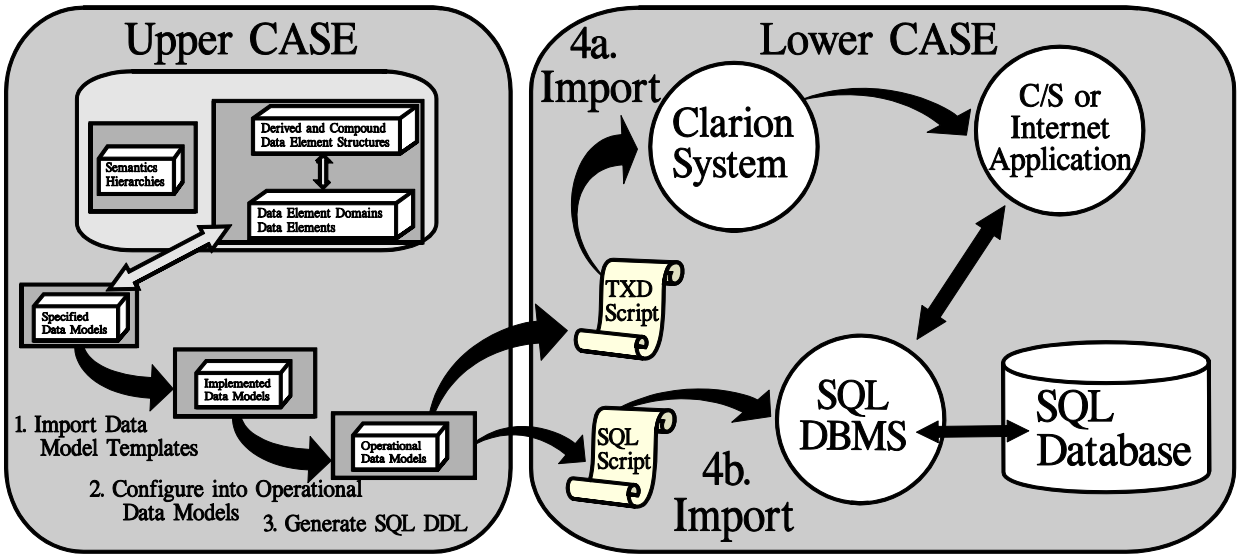

Figure 5. The role of the Metabase in an Enterprise Architecture Environment.

Not emphasized is the third objective because there are a number of operational/technical metadata repositories that are more than adequate. The first objective, that is, metadata that if unaddressed causes 95% of all database and business information systems errors. Focus should be on that objective.

Metadata repositories that store just operational or technical metadata miss almost all the true value of metadata to the enterprise. Clearly, if the focus is solely on the optimization of the past, enterprises will never be able to plan, see, and accomplish the future. It is through the exposition, optimization, and recasting of our missions, organizations, functions, database objects, databases and information systems that the future will be unfolding in ever better forms. Consequently, the database design of a metadata repository should closely match the metadata requirements.

There is also a very compelling business rational for a metabase. The data contained in the metabase is parallel to the financial data contained in a company's finance. In short, because well ordered financial data is essential to a well-ordered business then well-ordered metadata is essential to information technology.

Just having a methodology, a metadata repository, and well-defined product artifacts do not mean that there is integrated and non redundant metadata. It is delusional to believe otherwise. If multiple databases and business information system projects develop their own metadata in isolation without achieving integration and non redundancy then there will just be an increase of stove pipes, but this time, the stove pipes will be of metadata.

The overall requirements, functions, and processes of any metabase system closely follow the sequence and content of the architectures that are constructed. Any quality metabase must however be able to build these architectures independently and then have facilities to reconcile the inevitable disconnects.

In support of providing more specificity to a quality metabase system, the section briefly describes the key functional modules that must exist and then an overall list of functional requirements that the metabase system must satisfy.

Over the past 20+ years, quality metabase systems have always increased productivity, increased quality, lowered costs and lowered risks. Additionally, when metabase systems satisfy the requirements of this section, and are built through code generators, they have an overall negative cost to the enterprise. That, is, it always costs more to be without metabase systems than to have them.

8.0 Enterprise Architectures Way Ahead

The way ahead involves answering the following questions:

- Do you want to increase productivity?
- Do you want to increase quality?
- Do you want to decrease cost?

- Do you want to decrease risk?

If the answers are yes, and it is certain that they are, then time and effort must be invested in establishing a knowledge worker environment. Is the Knowledge Worker Framework set out in this paper perfect? Certainly not. Is the set of architectures identified in this paper the definitive and complete list? Certainly not. The Knowledge Worker Framework and the architectures represent a good starting place. Waiting for perfection means never starting.

Key to the way ahead is to embrace the "meta world." That is, get comfortable in designing and building tools that make knowledge work easier, cost effective, repeatable, reliable, integrated, and non redundant.

The cost imperative for getting highly efficient knowledge workers is compelling. For example, in the 1960s, computers (32K words and 12 tape drives with no disks) cost $650 an hour to operate. Monthly staff costs were $650. Computers were 440 times more expensive to than were staff. Clearly then, staff was the most common and the least expensive resource.

Large quantities of staff were the solution to the overall computer hardware costs. Days or weeks between program compiles was common. No more. Today, very sophisticated and profoundly powerful computers (2 gig memory, 210 Gig of 15K RPM SCSI disk drives, and two dual-core 2.5 megahertz processors) can cost just $5,500 dollars to purchase. Over a 4 year period, that's $0.53 per hour to use. Staff costs are $125 per hour for "journeyman" efforts. That means that now, a staff member costs about 250 times more than the very powerful computer used by that staff member. Overall, that's a ratio reversal of 110,000, which is a number too big to comprehend, but stark nonetheless.

It should be abundantly clear then that it's the knowledge worker that needs to be optimized and made more productive. That is where the biggest payoff is. This paper therefore addresses the architectures that directly affect the quantity and quality of the work of the knowledge worker.

The bottom line to this paper is this: Maximize the working environment and productivity of the knowledge worker. Such a working environment is illustrated in Figure 5. In this environment, as the knowledge worker does his job via some methodology which is set within one or more frameworks, the knowledge worker is able to access one or more metabases to create, employ, or evolve work products that are of value on the project, or that will be of value for other projects. The Knowledge Worker Framework, the architectures, and the metabase, all described in this paper are a great place to start.

What are you to specifically do? Now that you have read the paper, take it and put it onto your manger's desk. Ask that it be read and acted upon. Your enterprise's very existence may depend on it.

Finally, it needs to be said, and remembered that this paper is not a theory paper. If it had been, it would have been written 25 or more years ago when all this was untested and unrefined. These materials have however been tested and refined by many industry and government organizations through many information technology projects. This paper therefore is the

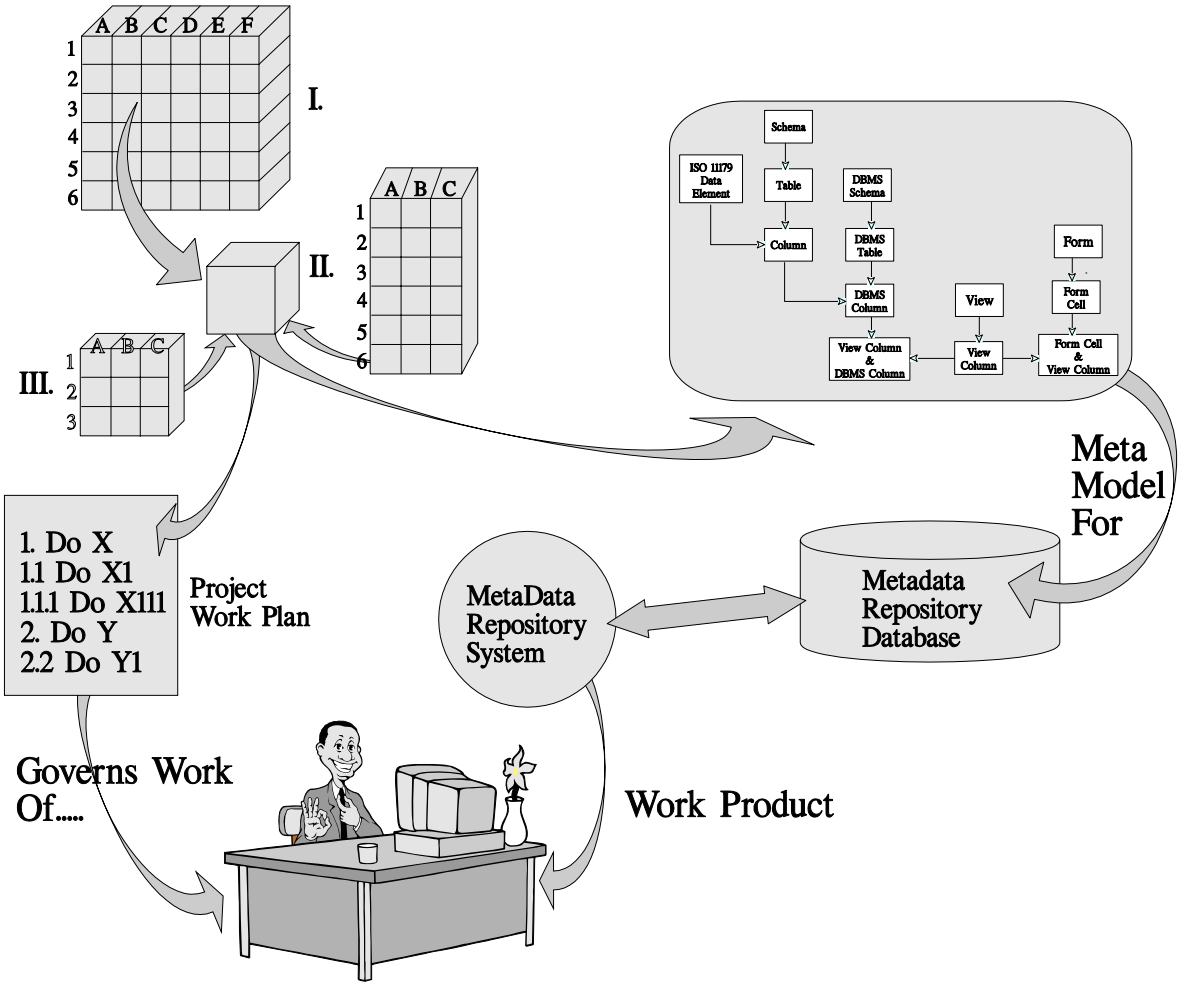

Figure 6. Unified operating knowledge worker environment with frameworks, metadata repository, and methodology.

consequence of the evolution and development of enterprise architecture and data management practices that have been conducted in industry and government since the early 1970s.

Every product in every section of this paper is fully detailed in other Whitemarsh books, and is fully supported by detailed methodology, courses, workshops, and of course, the metabase, metabase. In short, this paper is another integral component in the Whitemarsh Data Management Series.

9.0 References

The following references to Whitemarsh materials provide a more detailed exposition practical application of the significant content of this paper.

The following documents are available free from the Whitemarsh website:

Paper	URL
Whitemarsh Data Modeler, Architecture and Concept of Operations	http://www.wiscorp.com/MetabaseDataModelerArchitectueandConceptofOperations.zip
Metabase User Guides	http://www.wiscorp.com/MetabaseUserGuides.zip
Iterations of Database Design	http://www.wiscorp.com/iterations_of_database_design.pdf
Data Management Conferences	http://www.wiscorp.com/dama2002.zip http://www.wiscorp.com/dama2003.zip http://www.wiscorp.com/wrad2000.zip
Comprehensive Metadata Management	http://www.wiscorp.com/ComprehensiveMetadataManagement.pdf
Metabase Overview	http://www.wiscorp.com/Metabase.zip
Reverse and Forward Engineering Guide	http://www.wiscorp.com/MetabaseReverseAndForwardEngineeringUsersGuide.zip

The following documents are available for Whitemarsh Website Members. The URLs that follow provide descriptions of the pages. Members should log in and proceed to the appropriate page, e.g., Enterprise Database, find the book, paper, or course and perform the download.

Paper	URL
Data Management Program - Metadata Architecture For Data Sharing Data Management Program - Database Interface Architectures Data Management Program - Projects And Data-Asset Product Specifications	http://www.wiscorp.com/wwmembr/mbr_products_edb.html

Paper	URL
Knowledge Worker Framework Book Enterprise Database Whitepaper Metabase Overview WRAD Conference Talk Data Standardization Talk (DAMA)	http://www.wiscorp.com/DatabaseDesignInformation.html
Resource Life Cycle Analysis Database Objects	http://www.wiscorp.com/tdan.html
Work Breakdown Structures Database Project Work plan Templates Information Systems Development Methodology Phases 1 and 2 Whitemarsh Project Estimating Work plan Development	http://www.wiscorp.com/wwmembr/mbr_products_dp.html
Achieving Data Standardization Achieving Data Standardization Long Talk Achieving Data Standardization Short Talk Achieving Enterprise Wide Data Semantics Standardization Data Standardization Work Plan The Data Standardization Problem	http://www.wiscorp.com/wwmembr/mbr_products_dq.html
Data Management Program - Metadata Architecture For Data Sharing Data Management Program - Database Interface Architectures Data Management Program - Projects And Data-Asset Product Specifications Data Management Program - Work Breakdown Structures Knowledge Worker Framework Database Objects Managing Database - Four Critical Factors	http://www.wiscorp.com/wwmembr/mbr_products_edb.html
Data[base] Architecture Classes Guidelines for Data[base] Architecture Class - Data Warehouse	http://www.wiscorp.com/wwmembr/mbr_products_dd.html

Paper	URL
Iterations of Database Design	
Work Breakdown Structures	http://www.wiscorp.com/wwmembr/mbr_products_dp.html