Enterprise Architecture: The Issue of the Century

By
John A. Zachman

(This is the unedited version of an article that appeared in the March '97 issue of Database Programming and Design magazine.)

Introduction

The Information Age is unfolding just as predicted by many of the sociological prognosticators of this Century. Information issues are in everyone's mind and on multitudes of lips. It is hard to pick up a newspaper or current affairs magazine without seeing a feature on the Internet, Web Pages, E-Mail, Television Terminals or some other new technology. In fact, technology innovation is relentless and escalating and technology stocks continually drive the stock market to high after high. There is no field of human endeavor that is exempt from the onslaught of information technology.

At the same time, in the midst of this information frenzy, the Information Systems community is in a state of disarray ... down-sized, out-sourced, over-worked, package-replaced, stressed-out and decentralized. The credibility of I/S is in a steep decline. Everything we have built in the past 50 years (the "legacy"), and everything we are doing in the present to satisfy current demand, are perceived by management to be hopelessly inadequate.

In the last two years, Warren Selko of IBM CGI has interviewed CEOs and CIOs of 108 Fortune 500-type Enterprises and has uncovered substantial and unprecedented frustration. The stress levels are out-of-sight and CIO tenure is tenuous at best. One CEO said, "I would out-source that thing (I/S) in a heart-beat!" On a scale of 1 - 10, they rate the quality and timelines of information they get a 6 and the appropriateness between 3 and 4. Think about that ... what does a CEO do with anything less-than-perfect quality and timely information?

Furthermore, to a person, the CEOs declare that the biggest challenge facing the modern Enterprise is "change." A quick review of the history of all the known disciplines that deal with complex objects (things) reveals that change starts with the engineering descriptions of the things. There is no way to change hundred story buildings quickly (or safely) without starting with the building plans. There is no way to change Boeing 747's quickly (or safely) without starting with the product description. There is no way to change automobiles, computers, integrated circuits, oil refineries, battleships, telephone networks, programs, Enterprises, or any other complex thing quickly (or safely) without starting with the descriptive representations of the thing you want to change.

These issues of quality, timeliness and change are the conditions that are forcing us to face up to the issues of Enterprise Architecture. The precedent of all the older disciplines known today establishes the concept of architecture as central to the ability to produce quality and timely results and to manage change in complex products. Architecture is the cornerstone for containing Enterprise frustration and leveraging technology innovations to fulfill the expectations of a viable and dynamic Information Age Enterprise.

Although for decades we have known of the significant nature of the issue of architecture, it is only relatively recently that both our conceptual understanding and the technology and methods have allowed us to become practical about it. My opinion is, we are on the verge of seeing Enterprise Architecture "come into its own," and in the 21st Century, it will be the determining factor, the factor that separates the winners from the losers, the successful and the failures, the acquiring from the acquired, the survivors from the others.

Background

I came from the information strategy community in the early days and even by the late 1960's, we were quite competent to do information strategy. Although the strategy tools and the methods have improved substantially, the analytical process was quite well understood decades ago. Our problem was, we were having grave difficulties getting from strategy ... to implementation.

We knew that architecture had everything to do with bridging the gap between the strategy (expectations) and the implementations, and with establishing an Enterprise environment that was conducive to change. The problem was, we didn't know what architecture was.

In 1972, I moved out to the Los Angeles area, where I was doing a lot of Information Strategy work among air-frame manufacturing companies. Those of us who were working in this industry at that time were struck by the great similarities between what they were doing, manufacturing airplanes, *complex engineering products* ... and what we were trying to do, build Enterprise systems, *complex engineering products*. The more we learned about airplane manufacturing, the more we were convinced that Information Systems (Enterprise Engineering) was simply a different instance of the same generic, complex, product development and manufacturing process.

However, one notable difference was, the airplane manufacturers had clearly figured out what architecture was as it related to airplanes. They could successfully transit the architectural bridge between strategy and implementation. We could see physical manifestation of it. For example, they were able to produce very complex engineering products for a dynamic marketplace that were relevant to the marketplace. That was one of the problems we were having. We were trying to produce Enterprise systems, very complex engineering products, into a dynamic Enterprise, our "marketplace," that were relevant to our marketplace. Our problem was, by the time we could get the product (system) produced, it was no longer relevant to the marketplace (Enterprise.)

Another thing the air frame manufacturers were doing was building airplanes piece by piece, part by part, sub-assembly by sub-assembly, and when they get it all done, all the pieces, parts, sub-assemblies fit together into an airplane ... and the thing could actually fly.

That was another problem we were having. We were trying to build these Enterprise systems, piece by piece, program by program, system by system, application by application, and when we get them all done, they weren't fitting together all that well. And ... management was somewhat unrestrained in observing that the Enterprise wasn't "flying" too well as a result!

The third thing the air frame manufacturers could do was maintain the life of those very complex products over very long periods of time, in the face of dynamically changing marketplace and dynamically changing technology. It is not unusual to find an airplane relevant and flying for over 50 years. Our problem was, even if we could get the product (system) produced into the Enterprise such that it was relevant, we could not maintain its relevance for long periods of time. It tended to become obsolete very quickly and the attempts to maintain its relevance time-consuming and expensive. Further, the costs and time of maintenance approximate exponential increases with an increase in the number of systems implemented.

In any case, the air frame manufacturers had clearly figured out how to bridge the gap between the strategy and the implementation, and how to manage product change, that is, they had clearly figured out what architecture was as it related to airplanes. My idea was merely to learn what they had learned and interpret that lesson into a different domain, "systems," that is, the domain of Enterprises.

The Framework

There are several other sources, including the references cited in this article, that describe in some detail what I learned, not only in Engineering and Manufacturing, but also in Architecture and Construction where the conceptual structures are identical. The building and airplane metaphors are equally valuable, and in brief, from these older disciplines, I discovered that there is not simply a single architectural representation for a complex product. There is a set of representations. There are representations from different perspectives, or roles, being played in the process of producing the product. For example, there are representations of the end product from the perspective of the customer, or ultimate "owner;" from the perspective of the engineer, or "designer;" and, from the perspective of the manufacturing engineer, or "builder" of the product; along with some other perspectives.

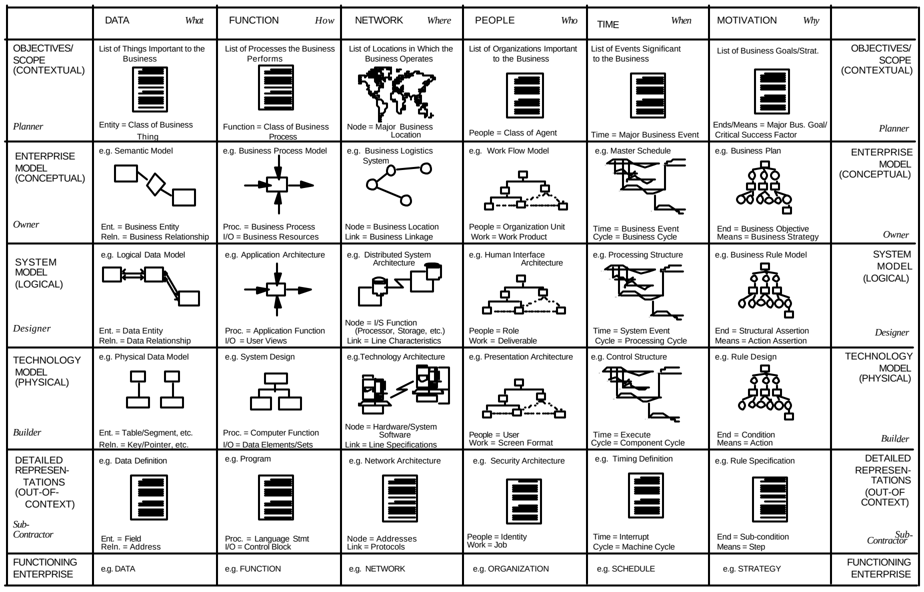
There are also intersecting representations of the different characteristics of the product. For example, there are representations of the material composition of the product from the perspective of the owner, from the perspective of the designer, from the perspective of the builder, etc. Likewise, there are intersecting representations of the function and the geometry of the product from the perspectives of the owner, designer and builder.

The breakthrough realization that ultimately resulted in the Framework as it exists today was the realization that the representations of the intersecting characteristics, that is “material,” “function,” and “geometry,” were actually descriptions of WHAT the product was made of, HOW the product worked and WHERE the components were located relative to one another. From that observation, it was obvious that a *comprehensive* description of the product necessarily would have to include descriptions of WHO does what relative to the product, WHEN do things happen and WHY are various product choices being made.

In total, I could identify five different perspectives, the Scope, Owner, Designer, Builder and what I have called the “Out-of-Context” (or, Sub-Contractor) views, over and above the end product itself. Also, I identified six characteristics, which I call “abstractions” including the What, How, Where, Who, When and Why product characteristics. I called these characteristics “abstractions” because, when describing an object, it is convenient to isolate a single characteristic at a time. Attempting to deal with all the characteristics at one time would result in such a complex depiction it would be incomprehensible, useless. It is a process of “abstraction,” of extracting out of the total, a more simplistic sub-set on which to focus for some particular exercise.

I simply took this generic framework of Perspectives and Abstractions that I derived through observation of the descriptive representations of physical products (namely airplanes and buildings) and applied it to an Enterprise to produce the Framework for Enterprise Architecture as illustrated in Figure 1.

A FRAMEWORK FOR ENTERPRISE ARCHITECTURE ™



	DATA *What*	FUNCTION *How*	NETWORK *Where*	PEOPLE *Who*	TIME *When*	MOTIVATION *Why*	
OBJECTIVES/ SCOPE (CONTEXTUAL) *Planner*	List of Things Important to the Business Entity = Class of Business Thing	List of Processes the Business Performs Function = Class of Business Process	List of Locations in Which the Business Operates Node = Major Business Location	List of Organizations Important to the Business People = Class of Agent	List of Events Significant to the Business Time = Major Business Event	List of Business Goals/Strat. Ends/Means = Major Bus. Goal/ Critical Success Factor	OBJECTIVES/ SCOPE (CONTEXTUAL) *Planner*
ENTERPRISE MODEL (CONCEPTUAL) *Owner*	e.g. Semantic Model Ent. = Business Entity Reln. = Business Relationship	e.g. Business Process Model Proc. = Business Process I/O = Business Resources	e.g. Business Logistics System Node = Business Location Link = Business Linkage	e.g. Work Flow Model People = Organization Unit Work = Work Product	e.g. Master Schedule Time = Business Event Cycle = Business Cycle	e.g. Business Plan End = Business Objective Means = Business Strategy	ENTERPRISE MODEL (CONCEPTUAL) *Owner*
SYSTEM MODEL (LOGICAL) *Designer*	e.g. Logical Data Model Ent. = Data Entity Reln. = Data Relationship	e.g. Application Architecture Proc. = Application Function I/O = User Views	e.g. Distributed System Architecture Node = I/S Function (Processor, Storage, etc.) Link = Line Characteristics	e.g. Human Interface Architecture People = Role Work = Deliverable	e.g. Processing Structure Time = System Event Cycle = Processing Cycle	e.g. Business Rule Model End = Structural Assertion Means = Action Assertion	SYSTEM MODEL (LOGICAL) *Designer*
TECHNOLOGY MODEL (PHYSICAL) *Builder*	e.g. Physical Data Model Ent. = Table/Segment, etc. Reln. = Key/Pointer, etc.	e.g. System Design Proc. = Computer Function I/O = Data Elements/Sets	e.g.Technology Architecture Node = Hardware/System Software Link = Line Specifications	e.g. Presentation Architecture People = User Work = Screen Format	e.g. Control Structure Time = Execute Cycle = Component Cycle	e.g. Rule Design End = Condition Means = Action	TECHNOLOGY MODEL (PHYSICAL) *Builder*
DETAILED REPRESEN-TATIONS (OUT-OF-CONTEXT) *Sub-Contractor*	e.g. Data Definition Ent. = Field Reln. = Address	e.g. Program Proc. = Language Stmt I/O = Control Block	e.g. Network Architecture Node = Addresses Link = Protocols	e.g. Security Architecture People = Identity Work = Job	e.g. Timing Definition Time = Interrupt Cycle = Machine Cycle	e.g. Rule Specification End = Sub-condition Means = Step	DETAILED REPRESEN-TATIONS (OUT-OF CONTEXT) *Sub-Contractor*
FUNCTIONING ENTERPRISE	e.g. DATA	e.g. FUNCTION	e.g. NETWORK	e.g. ORGANIZATION	e.g. SCHEDULE	e.g. STRATEGY	FUNCTIONING ENTERPRISE

John A. Zachman. Zachman International

In a brief article, it is impractical to elaborate all the steps of development and validation that have taken place over the years. However, Barbara von Halle, a regular contributor to Database Programming and Design, has devoted quite a number of her columns to discussions of the Framework. She has the advantage of being married to an Architect (as in Architecture and Construction of buildings) who could not only validate many of the structural assertions of the Framework, but also provide insight into the not-so-obvious process of doing actual architecture work. Barbara’s most recent article was “Architecting in a Virtual World” which documents many of the thoughts she shared at the recent Zachman Institute for Framework Advancement (ZIFA) Forum that was held in June ‘96 in Orlando.

Definition

So, what is “architecture?” My definition would be, “Architecture is that set of design artifacts, or descriptive representations, that are relevant for describing an object such that it can be produced to requirements (quality) as well as maintained over the period of its useful life (change).”

My contribution to the body of knowledge was simply identifying the generic logic structure that organizes, or classifies, the descriptive representations of complex objects, actually of *any* object, as in Figure 2.

	WHAT	HOW	WHERE	WHO	WHEN	WHY
SCOPE						
OWNER						
DESIGNER						
BUILDER						
SUB-CONTRAC-TOR						
PRODUCT						

Figure 2. The Framework, a generic classification scheme for descriptive representations of any object.

In the case of Enterprises, the definition of architecture would be “that set of descriptive representations (i.e. ‘models’) that are relevant for describing an Enterprise such that it can be produced to management’s requirements (quality) and maintained over the period of its useful life (change).”

Granularity

Since the logic of the Framework is generic, it can be used to classify the descriptions of anything, including: airplanes, buildings, automobiles, computers, Enterprises, or bicycles, or what have you. It could also be used to classify the descriptions of parts, sub-assemblies, components, sections, products or product families.

This latter set of examples is a special case in that the objects are contained within each other. They have a “bill-of-materials” type of relationship. One is a sub-set, or component, of another. In the fields of building architecture and product engineering, this component within component (bill-of-materials) issue is called the “granularity” issue. Each component has an architecture in its own right, but must “fit” within the architecture of the whole. That is, each component has a Framework of descriptive representations in its own right, but that component Framework must “fit” within the Framework of descriptive representations of the whole object (i.e. product, or Enterprise, etc.) being built. In such a granularity case, the architect/engineer must be meticulous to ensure that every sub-part is architected to fit within the super-part. That is, it must be structurally consistent (fit within the bills-of-material, column 1 representations), spatially consistent (fit within the drawings, or geometry, column 3 representations), and behaviorally consistent (fit within the motivation, or objectives, column 6 representations).

In the Enterprise manifestation of the same principle, the “architect” must be meticulous to ensure that every sub-part (that is, program, system, application, departmental implementation, etc.) must integrate into the Enterprise bill-of-materials (the Enterprise data models, column 1), the Enterprise geometry (the hardware and systems software

	WHAT	HOW	WHERE	WHO	WHEN	WHY
SCOPE	▨		▨			▨
OWNER	▨		▨			▨
DESIGNER	▨		▨			▨
BUILDER	▨		▨			▨
SUB-CONTRAC-TOR	▨		▨			▨
PRODUCT	▨		▨			▨

Figure 3. Models (shaded) sensitive to Enterprise-wide integration and therefore affected by granularity.

network, column 3) and the Enterprise objectives and strategies (Enterprise business rule models, column 6).

Therefore, if something less in scope than “the Enterprise” is being built (that is, if a program, a system, an application or any other gradation of departmental implementation is the scope), even when the Framework is being used as a guide for model-driven approaches to deliver quality results, the question that begs asking is, “are semantic discontinuities, network anomalies or rule inconsistencies being introduced into the Enterprise as a whole?” These may cause more Enterprise problems than the local benefits the implementation will realize. In many cases, steps may need to be introduced to maintain integrity within Enterprise scope of data, network and rules (columns 1, 3, and 6.)

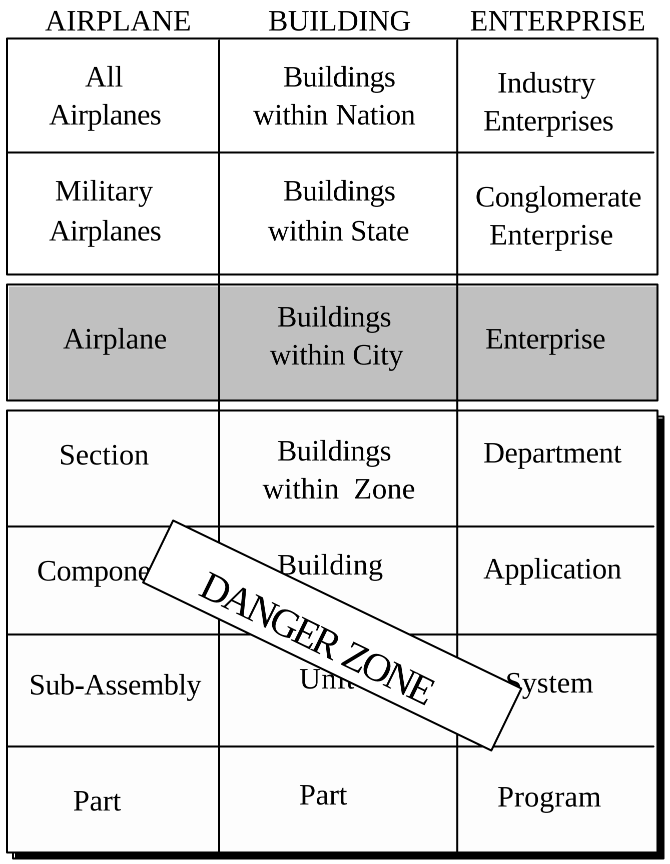
Figure 3 identifies the Framework models that are sensitive to integration and therefore are impacted by the issue of granularity.

This raises the question, “what constitutes the boundaries of ‘the Enterprise?’”
Figure 4 gives some examples of granularity in airplanes, buildings and Enterprises and it appears to me, after some extensive consideration, that the level in the hierarchy in which the object functions as a stand-alone, self-contained unit is the level that constitutes a natural boundary for integration. In general, a self-contained “Enterprise” would be defined by a set of business functions and assets, all integrated in support of a common mission or set of objectives. This has been called at various times, a Division, Strategic Business Unit, a profit center, a public sector “agency,” an “Enterprise.”

Clearly, there are exceptions and I believe that the choice is made on the basis of manageable scope versus jurisdictional authority. That is, the natural integration boundary is the level at which the next higher level exerts no (or little, or only partial) influence on standardization, or integration, or control issues. For example, when I talk about the Framework for Enterprise Architecture, I mean architecture at the level of the “Enterprise” as indicated by the shading in Figure 4, not simply a system, an application or some organizational sub-set of the Enterprise. Conceptually, this is the easiest, “cleanest” definition of “Enterprise” that constitutes a natural integration boundary. The disintegration “danger zone” lays anywhere below the natural integration boundary as shown in Figure 4.

Granularity and the Enterprise

This issue of granularity is of paramount importance to today’s Enterprise because it has everything to do with the frustrations the Enterprise is feeling with regard to the legacy



AIRPLANE	BUILDING	ENTERPRISE
All Airplanes	Buildings within Nation	Industry Enterprises
Military Airplanes	Buildings within State	Conglomerate Enterprise
Airplane	Buildings within City	Enterprise
Section	Buildings within Zone	Department
Compone	Building	Application
Sub-Assembly	Uni	System
Part	Part	Program

DANGER ZONE

Figure 4. Architecture Granularity

and our ability to satisfy current demand. We have been operating at the lowest levels of architectural granularity for the fifty years of history of data processing. Regardless of our intention and even if we have been employing the best architectural techniques by using model driven approaches to systems development, if the implementation was architected at the program, system, application or departmental levels, we inadvertently or sometimes even deliberately introduced substantial discontinuity and redundancies into the Enterprise data, the network and the business rules (columns 1, 3 and 6 models.)

Referring to these redundancies and discontinuities, in a recent presentation (at the Barnett "All About Information Resource Management '96" Conference, July 20-24 in Snowmass, Colorado) Larry English of Information Impact International observed that "70 percent of all computer printouts were used to re-enter data into other databases." "One company reported that 80 - 90 percent of developers' time was devoted to 'maintenance' and of that 'maintenance,' 60 - 70 percent was devoted to maintaining interfaces, copying and transforming data from database to database." "Another company reported expending $100 million per year in patching programs and fixing errors in data, created when passing data from one system to another."

At the Zachman Institute for Framework Advancement (ZIFA) Forum, June '96 in Orlando, Bill Smith of William G. Smith Associates observed "70 percent of the lines of code in your company that you are maintaining are doing nothing but moving data from system to system, file to file" ... "40 percent of the machine cycles are expended moving data, doing no productive work" ... one bank he has worked with was storing customer data redundantly in 129 files ... that they knew of. "Statistically, the average data fact is stored 10.8 times redundantly ... this is neither smart nor cheap."

Doug Erickson of Data, Applications: Technology Associates, in a soon to be published article cites estimates that "between 20 percent and 40 percent -- one estimate puts the figure at 50 percent -- of all labor costs in the U.S. is dedicated to gathering, storage, retrieval, reconciliation and reporting of the information used to run an enterprise." Any of this that is redundant is likely unnecessary, causing information "float," driving up the labor costs of reconciliation and damage control by someone outside of I/S but within the Enterprise.

Data Warehouse

The intense world-wide preoccupation with "data warehouse" should have been highly predictable as somehow, we had to find an approach to compensate for this historical lack of Enterprise-wide data architecture. The data in the existing systems, the "legacy," is so discontinuous, so inconsistent, so incorrect, so redundant, it borders on useless for management purposes as cited in the CEO survey earlier. The legacy data may serve to get the transactions processed, but when validity of its values and its meaning become vital to understand what is happening in or to the Enterprise, it is seriously deficient and therefore contributes substantially to the frustration the Enterprise is feeling toward I/S. The creation of the data warehouse (i.e. creating the integrated semantic structure) and the extracting, transforming, cleansing, integrating and distribution of the data is simply our after-the-fact effort to redeem a very tenuous, Enterprise-frustrating, legacy situation. I do not want to minimize the very significant advances in decision support systems, analytical processing, data mining, and so on that the nearly universal focus on data warehouse has brought to the information discipline, but setting these universal, industry-wide advances aside for a moment, the expense of building a data warehouse in a given Enterprise is substantial. Sid Adelman of Sid Adelman & Associates in a recent presentation observed that the Meta group estimates the cost of a single data warehouse implementation project runs around $3 million, and that is for a single, *initial* implementation, nowhere near the scope of providing integrated views for an entire Enterprise. By far and away, the major effort in data warehouse projects lies in the "reverse engineering," (digging through the legacy and the archived files, element by element, trying to figure out the meaning and how it is polluted), and then cleaning up the errors in the actual instance data in the files.

Had an Enterprise data architecture and some respect for its sanctity been in place, there would be no need for this reverse engineering, reconciliation and cleansing process as we know it today. It would be relatively straight-forward to load the data into a specialized data base, or construct a different schema, or to create multiple dimensions, or to search for visually obscure patterns, etc. I would suggest that the expenditure on data warehouses is somewhat after-the-fact indicative of the value of data architecture in the first place, and before an Enterprise gets done with building data warehouses, there is going to be *a lot* of money spent that could have otherwise been avoided.

Further, I would suggest that if the issues of Enterprise Architecture are not encompassed in any current data warehouse implementation, the ultimate result will be *increased* Enterprise frustration and the expenditure will be perceived as dissipated as the implementation will simply be one more un-architected, redundant, legacy file, with a new name, "Data Warehouse."

With reference to this architectural integrity issue, Bill Inmon of Pine Cone Systems said at the Barnett, Snowmass conference, "I never said to build a mess on top of a mess! I said this is the opportunity to clean up the mess!"

The Year 2000

What about the year 2000? The Gartner Group estimates that $400 billion will be spent world-wide. I have seen estimates of anywhere between $1 and $4 per line of code in every Enterprise to transition successfully into January 1st, 2000. Once again, this is an after-the-fact attempt to compensate for the lack of data architecture, at the most pedestrian perspective of data architecture, data element definition, a column 1, row 5 architectural over-sight for one single, solitary data element! Not only did we design into that data element inconsistencies and discontinuities, we didn't retain any of the models to serve as a base-line for managing change.

How many other data elements are out there that are ill-designed, unmanaged and subject to change? I realize the ubiquity of the "date" data element, but there are a lot of other data elements like customer data, product data, part data, geographic data, ZIP code, European currency data, SKU number, etc., and some likely substantial number of them are candidates for pervasive change with the same kind of cost and Enterprise show-stopping implications.

What about changes to any of the other 29 types of architectural models defined by the Framework?! What happens when we want to make a business process change, or a business rule change, or a screen change, or a work flow change, or an operating system change, or a database management system change, etc.? What happens to the Enterprise when we (or, it) simply can't make the desired change by the time it is required, or within the limits of available capital?

Clearly, the modern Enterprise cannot bear the kind of cost and inflexibility induced by architectural deficiencies and survive in a highly competitive, dynamic, global marketplace.

The Network and Business Rules Models

I have not even begun to address the architecture problems associated with the network (column 3) and the business rules (Column 6). For one thing, the state-of-the art is still somewhat limited in business rule modeling (column 6) and virtually all of the business strategies, business rules, conditions, triggers, etc. are currently expressed in either the semantic (data) structures or procedural code. However, I guarantee, when competitive pressures begin to force attention on the business rule models to be responsive to the changes in the marketplace, it will be far easier to address them if the existing data and process architectural models are defined and being managed!

Regarding the network models, the costs of the plethora of vendors, systems, operating systems, releases, versions, bridges, interfaces, training, changes, failures, down-time, redundancy, etc. are considerable. And, the internet/intranet is no panacea. It looks to me like the complexity and costs of installing, maintaining, operating and changing the "nets" is enormous and clearly, the issue of architecture does not go away, in fact, I submit, it becomes *even more* important.

I have not expended the effort to locate all the original research references that substantiate the numbers and statistics I have referenced in this article. However, the references are almost irrelevant as these conditions likely exist in every Enterprise, within plus or minus 4 percent, a standard deviation. Therefore, in any given Enterprise, a modest amount of effort and determination could easily produce Enterprise-specific numbers and statistics if validation was necessary to precipitate action.

Architecture Is Free

Top management in every Enterprise is under heavy pressure to improve performance. To quote Doug Erickson once again, "In 1967, 40 - 50 percent of the cost of a product was direct labor cost. Today, the direct labor cost ratio is often as low as 15 percent." Where do you think management is going to get any more major chunks of cost reduction? It looks to me like these enormous costs of architectural discontinuities and redundancies are now "low hanging fruit" just waiting to be picked!

Doug argues strongly that just like in the manufacturing quality initiatives, the Enterprise Architecture initiatives are proving that doing it right (that is, architected for the Enterprise as a whole) prove to be faster and cheaper. Just like "quality is free," *Architecture is free*. "Every day that you don't build something that's redundant you are starting to generate benefit, and every time a piece of data is reused, the enterprise gets a 100 percent return on its investment."

One thing is abundantly clear from the numerical and statistical references earlier, a modest up-front investment has monumental downstream pay-back implications. If Doug Erickson's experience proves replicable, the up-front investment will also deliver implementations faster and cheaper to satisfy current demand as well as reduce the enormous costs and frustrations of the legacy paradigm.

New Enterprises have the luxury of the opportunity for doing it right. The older Enterprises have the necessity of doing it over. Bill Smith argues that cost of replacing the legacy is *not even an issue*. The new entrants and competition are forcing the legacy systems to be re-written. The only question is, how long will it be, and are they going to be architected ... or just re-written? If they are architected, architecture would be (virtually) free because if they are merely re-written, they will be re-re-written every time the new entrants, competition or market demand forces new changes into the Enterprise, which is happening with increasing frequency.

A Little Philosophy

To wax philosophical for a moment, as I grow older, I have come to realize that life is a series of decisions. There are good decisions and bad decisions, urgent decisions and important decisions ... but every decision has either short term or long term implications. There is nothing wrong about short term or long term decisions except, it is abundantly clear, if you exclusively make short term decisions and ignore the long term implications, there comes a time when you will live to regret it. You are going to “pay later.”

We can observe this phenomena socially, politically, spiritually, professionally, culturally, in every dimension of life. Tragically, we live in a time of strong cultural inclination for immediate gratification, fueled by the requirement to perform for “the bottom line” at the end of every quarter. It is little wonder that ALL political parties are crying out for VALUES. Values require INVESTMENT, and investment requires INTELLECTUAL CAPITAL. It costs little more in time or money to think before we act, but it does cost responsibility and thought. “Eat, drink and be merry, for tomorrow we die,” or “Doing what feels good” is not going to carry the day in a dynamic, Information Age environment when you are motivated toward any longevity at all.

If we want the luxury of satisfying immediate needs without an enormous sacrifice in downstream capabilities, we must make an investment in infrastructure; socially, culturally, spiritually, professionally and ... architecturally. Increasing flexibility and reducing time to market is not going to happen by accident or through one more technology acquisition or one more package or one more application implementation. It will only happen because of a responsible and intellectual investment in this case, in developing and maintaining Enterprise Architecture, to deliver quality information, in fact, to produce a *quality Enterprise.*

The issue of Enterprise Architecture is critical to the very survival of every Enterprise of any substance in the moderately near future, certainly by the early years of the 21st Century, and it is imperative that those of us in the information profession facilitate Enterprise assimilation of these monumental concepts.

My opinion is, we are on the verge of seeing architecture “come into its own” and in the 21st Century it will be the determining factor, the factor that separates the winners from the losers, the successful and the failures, the acquiring from the acquired, the survivors from the others. Today is not too soon to get started!

The Age of Enterprise Architecture is here.

References

1. "A Framework for Information Systems Architecture." John A. Zachman. IBM Systems Journal, vol. 26, no. 3, 1987. IBM Publication G321-5298. 914-945-3836 or 914-945-2018 fax.

2. "Extending and Formalizing the Framework for Information Systems Architecture." J.F. Sowa and J. A. Zachman. IBM Systems Journal, vol. 31, no. 3, 1992. IBM Publication G321-5488. 1-800-879-2755.

John A. Zachman
Zachman International
2222 Foothill Blvd. Suite 337
La Cañada, Ca. 91011
818-244-3763 (Phone and Fax)
johnzachman@compuserve.com