

# The Framework for Enterprise Architecture Getting Beyond the “Legacy”

by  
**John A. Zachman**  
© Copyright 1995 Zachman International




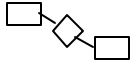


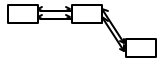

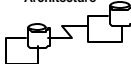


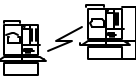



In the early '80's, there was little interest in the idea of Enterprise Reengineering or Enterprise Modeling and the use of formalisms and models was generally limited to some aspects of application development within the Information Systems community. The subject of “architecture” was acknowledged at that time, however, there was little definition to support the concept. This lack of definition precipitated the initial investigation that ultimately resulted in the “Framework for Information Systems Architecture.” Although from the outset, it was clear that it should have been referred to as a “Framework for *Enterprise* Architecture,” that enlarged perspective could only now begin to be generally understood as a result of the relatively recent and increased, world-wide focus on Enterprise “engineering.”

The Framework as it applies to Enterprises is simply a logical structure for classifying and organizing the descriptive representations of an Enterprise that are significant to the management of the Enterprise as well as to the development of the Enterprise's systems. It was derived from analogous structures that are found in the older disciplines of Architecture/Construction and Engineering/Manufacturing that classify and organize the design artifacts created over the process of designing and producing complex physical products (e.g. buildings or airplanes.)

The Framework graphic in its most simplistic form depicts the design artifacts that constitute the intersection between the roles in the design process, that is, OWNER, DESIGNER and BUILDER; and the product abstractions, that is, WHAT (material) it is made of, HOW (process) it works and WHERE (geometry) the components are, relative to one another. Empirically, in the older disciplines, some other “artifacts” were

observable that were being used for scoping and for implementation purposes. These roles are somewhat arbitrarily labeled PLANNER and SUB-CONTRACTOR and are included in the Framework graphic that is commonly exhibited. The Framework, as it is applied to an Enterprise, depicting Enterprise design artifacts (models,) using Enterprise terminology appears below.

### ENTERPRISE ARCHITECTURE - A FRAMEWORK

	DATA <i>What</i>	FUNCTION <i>How</i>	NETWORK <i>Where</i>
OBJECTIVES/ SCOPE  <i>Planner</i>	List of Things Important to the business  ENTITY = Class of Business Thing	List of Processes the Business Performs  Process = Class of Business Process	List of Locations in Which the Business Operates  Node = Major Business Location
ENTERPRISE MODEL  <i>Owner</i>	e.g. "Semantic Model"  Ent = Business Entity Rein = Business Relationship	e.g. "Business Process Model"  Proc = Bus Process I/O = Bus Resources	e.g. "Business Logistics System"  Node = Business Location Link = Business Linkage
MODEL OF THE INFORMATION SYSTEM  <i>Designer</i>	e.g. "Logical Data Model"  Ent = Data Entity Rein = Data Relationship	e.g. "Application Architecture"  Proc = Application Function I/O = User Views (Set of Data Elements)	e.g. "Distributed System Architecture"  Node = I/S Function (Processor, Storage, etc) Link = Line Characteristics
TECHNOLOGY MODEL  <i>Builder</i>	e.g. "Physical Data Model"  Ent = Segment/Row/etc. Rein = Pointer/Key/etc.	e.g. "System Design"  Proc = Computer Function I/O = Screen/Device Formats	e.g. "System Architecture"  Node = Hardware/Systems Software Link = Line Specifications
DETAILED REPRESENTATIONS  <i>Sub-Contractor</i>	e.g. "Data Definition"  Ent = Field Rein = Address	e.g. "Program"  Proc = Language Statement I/O = Control Block	e.g. "Network Architecture"  Node = Address Link = Protocol
FUNCTIONING SYSTEM	e.g. DATA	e.g. FUNCTION	e.g. NETWORK

From the very inception of the Framework, some other product abstractions were known to exist because it was obvious that in addition to WHAT, HOW and WHERE, a complete description would necessarily have to include the remaining primitive interrogatives: WHO, WHEN and WHY. These three additional interrogatives would be manifest as three additional columns of models that, in the case of Enterprises, would depict: WHO does what work, WHEN do things happen and WHY are various choices made. The state of the art in terms of modeling formalisms, as well as the inclination to devote energy to produce these additional artifacts is still somewhat limited, certainly in the case of Enterprises. The first three columns of models are quite adequate to draw some substantive conclusions, and therefore, in the interest of simplicity, the "other three columns" are being set aside for purposes of this discussion.

The older disciplines of Architecture and Manufacturing have accumulated considerable bodies of product knowledge through disciplined management of the "product definition"

design artifacts. This has enabled enormous increases in product sophistication and the ability to manage high rates of product change over time. Similarly, disciplined production and management of “*Enterprise* definition” (i.e. the set of models identified in the Framework for Enterprise Architecture) should provide for an accumulation of a body of *Enterprise* knowledge to facilitate enormous increases in *Enterprise* sophistication and accommodation of high rates of *Enterprise* change over time.

Although the Framework for Enterprise Architecture is an application of Framework concepts to Enterprises, the Framework itself is generic. It is a comprehensive, logical structure for descriptive representations (i.e. models, or design artifacts) of any complex object and is neutral with regard to the processes or tools used for producing the descriptions. For this reason, the Framework, as applied to Enterprises, is helpful for sorting out very complex, technology and methodology choices and issues that are significant both to general management and to technology management.

For example, out of the context of the Framework, it becomes clear why management and the users are in many cases so frustrated with the current inventory of existing applications, “the legacy.” The frustration clearly is not because there are too many applications. Or too few. Or because they are not strategic enough. Or because they are mainframe, or hierarchical, or COBOL, or whatever. The “legacy” is not a technology problem. It is an *architecture* problem.

The “legacy” frustration arises from three fundamental, architectural deficiencies.

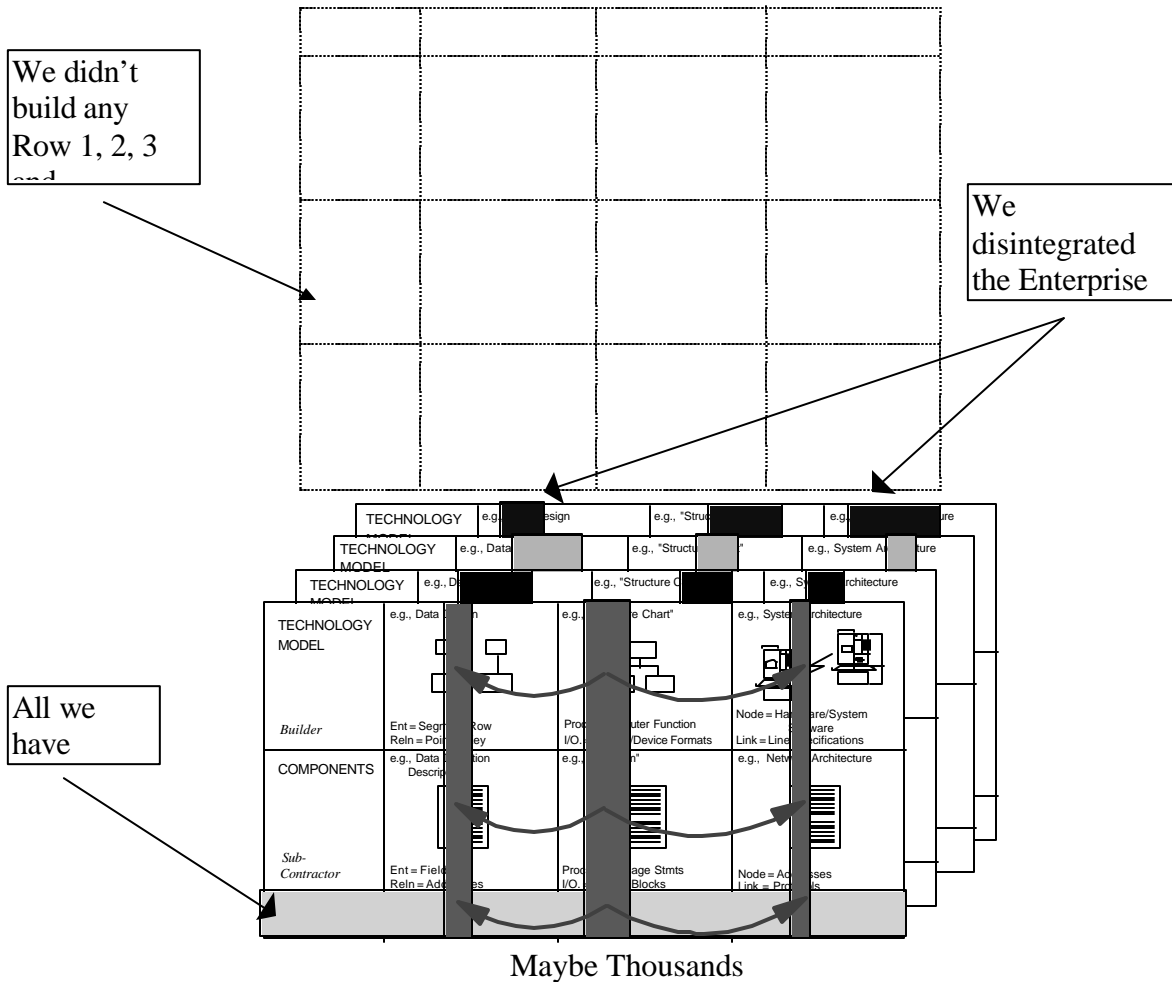
First, in general, Rows 1, 2 and 3 models were seldom built ... and in many cases, Row 4 models were not even built. The models which are not explicitly produced are implicitly assumed by default, and a lot of assumptions were made which have since proven to be, or have become, erroneous. Furthermore, if the Rows 1 and 2 models are only in the minds of management, that is, *implicit*, they can change as fast as Management can change their minds. On the other hand, the Row 6 implementations by definition are *explicit*, “poured in concrete,” resembling hundred story buildings. There simply is no way to keep the Row 6 reality in synch with the Row 2 perceptions without explicit formalizations and configuration control of the intermediate Row models as is done for physical products.

Second, the functional aspect of the systems was optimized at the expense of the data and the network for implementation expediency, that is, the data and hardware/systems software were tailored to the application and therefore disintegrated with regard to the Enterprise. Semantic and network discontinuities were created such that now, files can’t be related nor nodes connected and as a result, “maintenance” of the “legacy” now consumes the “lion’s share” of the I.T. resources. Worse, the energy of the *Enterprise* is consumed attempting to resolve the rule conflicts and overcome the communication complexities.

Third, the Enterprise has simply changed around the existing applications and these applications were built under the assumption that nothing would ever change. This is

clear because the only application models that are left in many cases are those that are machine readable ... Row 6. If anything changes in Rows 1, 2, 3, 4, or 5 above, those models in which the change occurs have to be “reverse engineered” from whatever information remains in Row 6. Some people liken this process to Archeology, that is, it is very costly and there is questionable confidence in its accuracy or even “do-ability.”

### SOURCES OF “LEGACY” FRUSTRATION



Now, the message to current Enterprise management as well as application developers is quite poignant out of the context of the Enterprise Framework. *If these architectural issues are not being explicitly addressed* in current Enterprise engineering and/or systems development projects, that is:

- a. if Rows 1, 2, 3, and 4 models are not being formally defined,
- b. if steps are not taken to preserve Enterprise-wide integration of Columns 1 and 3 models, and

- c. if all models produced are not being retained for change management purposes,

then, we are clearly *building more “legacy.”* We may well gain some temporary respite from the stress that is inherent in the existing inventory of systems, but the frustration and aggravation, not to mention the cost and complexity of maintaining the “legacy” and *the Enterprise’s inflexibility and unresponsiveness to change*, is bound to intensify as the Enterprise presses on into the dynamic, turbulent, “Information Age.” All the innovative technology “solutions” in the world including very good things like Data Warehouse, Object-Oriented, Client-Server, Parallel Processing, etc., etc. serve only to help build *more “legacy” faster.* We have to satisfy short term demand, but at the same time, it becomes obvious from looking at the “legacy” in the context of the Framework, if we ever intend to be free from the “legacy” problem, ***WE MUST FIRST ADDRESS THE ENTERPRISE ARCHITECTURE ISSUES!!***

The preceding discussion is an example of how the Framework for Enterprise Architecture can be used as a thinking tool to help understand why an Enterprise gets so frustrated with the “legacy” and to help develop a strategy to avoid adding to the “legacy” frustration in the future. Similarly, the Framework can be employed as a thinking tool to help understand a variety of complex issues and to develop deliberate and enduring strategies for creating flexible, agile, modern-day Enterprises designed to accommodate the vagaries and capriciousness of the “post-industrial” society.

Enterprise Architecture is the essential Enterprise challenge of the 21st Century Enterprise.

#### *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*