Excerpts From ...

The Zachman Framework For Enterprise Architecture:

Primer for Enterprise Engineering and Manufacturing

By
John A. Zachman

This document was prepared as a response to the OMG BRWG's RFI

Key Points

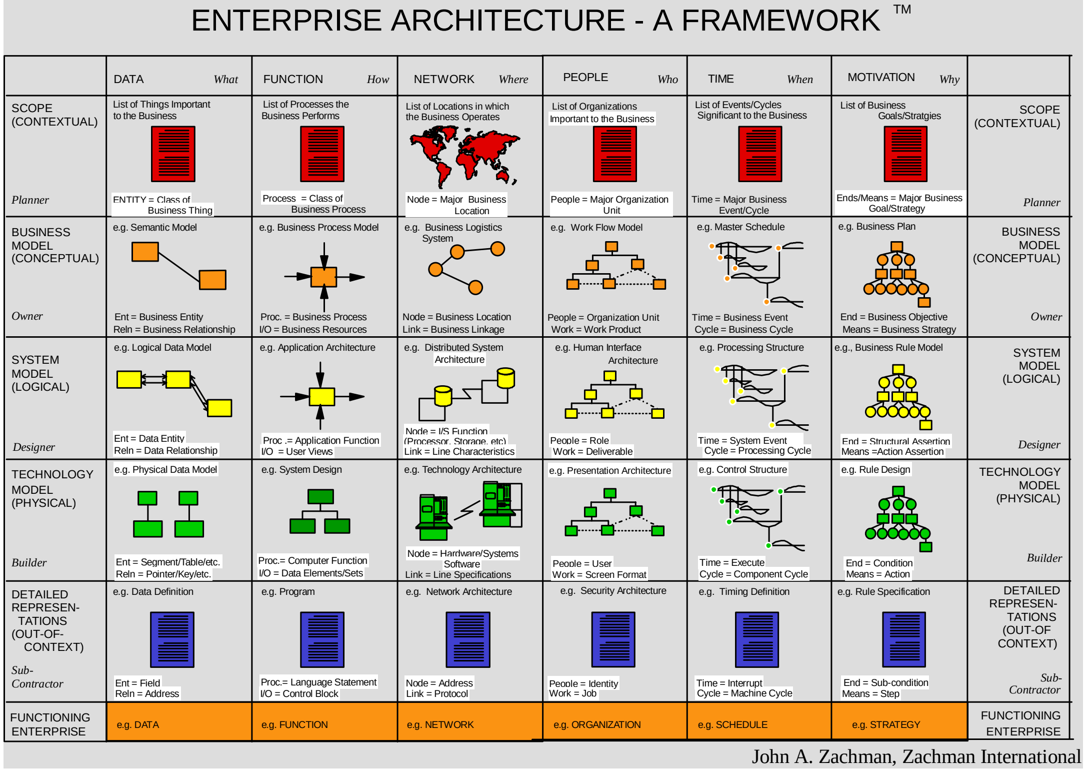
1. Seven thousand years of human history would establish that the key to **complexity and change** is Architecture. If it (whatever it is) gets so complex that you can't remember everything all at the same time, you have to write it down (Architecture). Then, if you want to change it (whatever it is), you start with what you wrote down (Architecture), the baseline for managing change. The reasons for doing Enterprise Architecture are, in the Information Age, it is the Enterprise that is getting complex and the Enterprise that is changing. (In the Industrial Age, it was the Product that got complex and the Product that had to change.)
2. The Framework for Enterprise Architecture (the "Zachman Framework") is a normalized schema, one (meta) fact in one place. That is what makes it a good analytical tool. Don't add or change the Rows or Columns or you will denormalize it and it will cease to be a good analytical tool. The Framework is a semantic structure. It implies **nothing** about implementation processes (methodologies) or tools whether they are top-down, bottom-up, left-to-right, right-to-left, or where to start.
3. The Framework can be used to help you think about (analyze) any thing or any Enterprise or portion thereof. The broader you define the analytical target, the better leverage you are going to get on integration, reusability, interoperability, etc., etc but the more complex the analysis. Conversely, the narrower you draw the boundary of the analytical target, the simpler the analysis but the less leverage you are going to get on integration, reusability, interoperability, etc., etc. If you draw the boundary beyond your jurisdictional control, you can no longer declare the models, you will have to negotiate the models. If you draw the boundary more narrowly than your jurisdictional control, you will disintegrate your Enterprise, that is, you will build a "legacy."
4. Once you get data manufactured, that is, designed and implemented in a database, there is no way to change the meaning of the data. After-the-fact attempts to post-integrate are okay, but you can only integrate (interface) cosmetic anomalies. If you only want to change the name or format or, if you only care about an individual record and don't care about its structure (meaning), you are home free. But, it is like putting lipstick on a pig … you can make it look good. The only way to change the meaning is scrap and rework.
5. You don't have to build out all of the models defined by the Framework, Enterprise-wide at excruciating level of detail before you can get to implementation. However, you have to remember that whatever slivers (vertical or horizontal) of whatever Cells you are not building (making explicit), you are making assumptions about, that is, you are assuming risk, risk of scrap and rework.
6. You don't have to build Enterprise-wide models in order to implement but you'd better pay attention to Enterprise-wide models in Columns 1, 3 and 6 because after you get systems implemented (e.g. the legacy) and the data doesn't mean the same

thing across the scope of the Enterprise, the network is fragile and costing a fortune to keep up 24 x 7 and the Objectives/Strategies (i.e. Business Rules) cannot administered consistently across the Enterprise, Management is going to be frustrated. After the systems get implemented, the only way to fix these kinds of problems short of cosmetic interfacing is scrap and rework.

7. If you are not observing the engineering design principles as related to the primitive Cell models, you are not going to realize the engineering design objectives of alignment, integration, reusability, interoperability, flexibility, reduced time-to-market, etc., etc., etc.
8. Until you have some (primitive) models stored somewhere in such a fashion that you can find them and reuse their components, you are, by definition, making-to-order, a "waterfall." It is only a matter of how wide or narrow the waterfall is and how many times you iterate through it enroute to implementation. That is, you are never going to appreciably reduce time-to-market until you have something in inventory before you get the order. In manufacturing, this would be called "mass-customization", assemble-to- order.
9. If you are not building (and storing, managing and changing) primitive models, you are not doing Architecture. You are doing implementations.
10. Early numbers indicate that conservatively, taking Enterprise Architecture based approaches as compared to the traditional application development approaches produces implementations 10 **times** cheaper and 6 **times** faster. This is not due to some kind of magic. It is simply because in employing Enterprise Architecture, the idea is to engineer the Enterprise first, before you manufacture it (implement) whereas traditionally, we manufactured the Enterprise (implemented) before we had it engineered (e.g. the legacy).

Introduction to the Framework

The Framework for Enterprise Architecture is a two dimensional classification scheme for descriptive representations of an Enterprise. (Open Figure 1. The Framework for Enterprise Architecture). It was derived through observation of descriptive representations (design artifacts) of various physical objects like airplanes, buildings, ships, computers, etc. in which it was empirically observed that the design artifacts (the descriptive representations, the product descriptions, the engineering documentation) of complex products can be classified by the audience for which the artifact was constructed (the Perspective) as well as classified by the content or subject focus of the artifact (the Abstraction).



ENTERPRISE ARCHITECTURE - A FRAMEWORK TM

	DATA *What*	FUNCTION *How*	NETWORK *Where*	PEOPLE *Who*	TIME *When*	MOTIVATION *Why*	
SCOPE (CONTEXTUAL) *Planner*	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	List of Organizations Important to the Business People = Major Organization Unit	List of Events/Cycles Significant to the Business Time = Major Business Event/Cycle	List of Business Goals/Stratgies Ends/Means = Major Business Goal/Strategy	SCOPE (CONTEXTUAL) *Planner*
BUSINESS 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	BUSINESS 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 = Segment/Table/etc. Reln = Pointer/Key/etc.	e.g. System Design Proc.= Computer Function I/O = Data Elements/Sets	e.g. Technology Architecture Node = Hardware/Systems 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 Statement I/O = Control Block	e.g. Network Architecture Node = Address Link = Protocol	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

Framework Description

Perspectives of the Framework

Different perspectives are being represented over the process of engineering and manufacturing complex products. The descriptive representations of the product that are prepared over this process are designed to express concepts/constraints relevant to the various perspectives. That is, not only do the design artifacts depict the necessary engineering information, but they depict it in such a fashion that it is intelligible to the perspective (audience) for which they were created.

The principal Perspectives are easily identifiable including:

- The Owner's Perspective (Row 2) - the recipient (customer, user) of the end product, (e.g. airplane, house, Enterprise, etc.)

  These descriptive representations reflect the usage characteristics of the end product, what the Owner(s) are going to do with the end product, or how they will use it once they get it in their possession. This is the conceptual view of the end product ... whatever the Owner can think about relative to its use.

- The Designer's Perspective (Row 3) - the engineer, the Architect, the intermediary between what is desirable (Row 2) and what is physically and technically possible (Row 4).

  These descriptive representations reflect the laws of nature, the system, or logical constraints for the design of the product. This is the logical view of the end product. For Enterprises, this is the logical representation of the Enterprise which forms the basis for the white collar system, the record-keeping system, of the Enterprise as well as the basis for the design of the blue collar system, the material manipulation system for manipulating the tangible aspects of the Enterprise.

- The Builder's Perspective (Row 4) - the manufacturing Engineer, the General Contractor, the employer of some technical capacity for producing the end product.

  These descriptive representations reflect the physical constraints of applying the technology in the construction of the product.[1001]

Empirically, there are two identifiable additional Perspectives which include:

- A Scope Perspective (Row 1) – the context that establishes the universe of discourse, the inner and outer limits, the list of relevant constituents that must be accounted for in the descriptive representations (models) for the remaining Perspectives.

- An Out-of-Context Perspective (Row 5) - a detailed description that disassociates the parts or pieces of the complex object for manufacturing purposes.

  These Out-of-Context representations play a part in the transformation from the media of the design of the product to the media of the end product

[1001] The words, Conceptual, Logical, and Physical as used here DO NOT mean high level of detail, medium level of detail and excruciating level of detail. Conceptual means anything the Owners can think of that they want the Enterprise to do or be. Logical means the systematic, unconstrained mechanisms employed by the Designer for realizing the Owners' concepts. (Administratively, this is the basis for the record-keeping, white collar systems of the Enterprise. Materially, this is the basis for the resource manipulation, blue collar systems of the Enterprise.) Physical means the technology-constrained implementation of the Designers' unconstrained, logical systems.

> itself.[1002] For example, in physical products, like airplanes, the medium of the design is typically paper and ink, (or more recently, electronic), whereas the media of the end product itself is aluminum, titanium, composites, etc. The Out-of-Context artifacts are employed in this media transformation between the media of the design and the media of the end product. For Enterprises, these are the product specifications relating the technology constraints of Row 4 to the vendor products in which the technology constraints are materialized.[1003]

It is worthwhile noting that Row 6 of the Framework represents the physical manifestation of the end product itself. Although, technically, Row 6 is not Architecture because it is not a representation (it is the actual thing), it is useful to incorporate it into the Framework graphic as it completes the Architectural picture. For an Enterprise employment of the Framework, Row 6 represents the Functioning Enterprise - the end result of the Architectural process. The end object is to ensure that Row 6 represents what the Owners have in mind for the Enterprise at Row 2. Row 6 is the realization of Row 2.

This set of perspectives appears to be universal and is easily observed in Architecture for buildings, independent of geography, culture, language, politics, or technology. Thousands of years of precedence establish that presently, in every case, there are the Bubble Charts or sketches (Scope), Architect's Drawings (Owner's View), Architect's Plans (Designer's View), Contractor's Plans (Builder's View) and Sub-Contractor's Plans (Out-of-Context View) and finally, the building (end product) itself. (Open Figure 2. Analogous Concepts.)

[1002] All of the preceding artifacts depict the product in context. The out-of-context artifact depicts the product disassembled into pieces, so that it can be manufactured piece-by-piece and then all the pieces are assembled into the final (Row 6) product.

[1003] For an Enterprise, the Row 5 models are the vendor product specification employed in the transformation from the physical representation (Row 4) to the implementation, i.e., object code (Row 6.) In Column 1, the Row 5 specification would be the database management system product language specifications. For Column 2 the Row 5 specification would be the programming language product specifications, etc.

BUILDINGS	ENTERPRISES
DESIGN ARTIFACTS	ENTERPRISE ARCHITECTURE
"BUBBLE CHARTS" Gross Sizing, Shape, Spatial Relationships	OBJECTIVES/SCOPE Gross Sizing, Scope
ARCHITECT'S DRAWINGS Building as Seen by the Owner	MODEL OF THE BUSINESS Enterprise as Seen by the "Owner"
ARCHITECT'S PLANS Building as Seen by the Designer	MODEL OF THE "SYSTEMS" (the "Systematic Model") Enterprise as Seen by the "Designer"
CONTRACTOR'S PLANS Architect's Plans as Constrained by Nature and Available Technology (Building as Seen by the Builder)	TECHNOLOGY MODEL Systematic Model as Constrained by Available Technology (Enterprise as Seen by the "Builder")
SHOP PLANS Descriptions of Parts/Pieces	DETAILED REPRESENTATIONS Description of Parts/Pieces
FUNCTIONING BUILDING	FUNCTIONING ENTERPRISE

Figure 2. Analogous Concepts

This would suggest that these are the **primitives,** that is, they are all different, they vary independently and that this is the minimum set of Perspectives (they all must be present) and further, **comprehensive**, that is, additional perspectives are not necessary. This is a fixed list. It is not a hierarchy. (The suggestion that this is the minimum set presumes that you care about integrity between the initial intent specified in the Scope and Owner's perspectives of Row 1, 2 and the end result expressed in Row 6.)

Abstractions of the Framework

The Abstractions, the other dimension of the classification system, depict the independent variables that constitute a comprehensive depiction of the subject or object being described,[1004] including:

- *What* it is made of - the material composition of the object, the bill-of-materials- for Enterprises, the Thing (Data) Models (Column 1).[1005]

[1004] Rudyand Kipling, The Elephant's Child 1902.

[1005] Actually, calling Column 1 the Data Column is a misnomer. It should be called the Thing Column because all the cells are descriptive of the Things of the Enterprise. It is only at Row 3 that they become Data Models. However, if I labeled Column 1 the Thing Column, no one would have a sense of what kind of models to expect, as common usage does not include Thing Models, at least, not at the present time.

- *How* it works - the functional specification, the transformations - for Enterprises, the Process (or Function) Models (Column 2).
- *Where* the components are located relative to one another - the geometry, the connectivity - for Enterprises, the Logistics (or Network) Models (Column 3).
- *Who* does what work - the manuals, the operating instructions - for Enterprises, the People (or, Work Flow) Models (Column 4).
- *When* do things happen relative to one another - the life cycles, the timing diagrams - for Enterprises, the Time (or, Dynamics) Models (Column 5).
- *Why* do things happen - the ends/means - for Enterprises, the Motivation Models (Column 6).

I chose the word Abstractions for this dimension of the Framework because the subject (or object) being described tends to be so complex that it is impossible to take into consideration all of the interrelationships of all the various components (independent variables) all at one time.[1006] It is complicated enough dealing with a single variable (abstraction) at a time.

"What" Material Description	**"How"** Functional Description	**"Where"** Spatial Description	**"Who"** Operational Description	**"When"** Timing Description	**"Why"** Motivation Description
Structure (Things)	*Transform* (Processes)	*Flow* (Locations)	*Operations* (People)	*Dynamics* (Events)	*Motivation* (Strategies)

Figure 3. Framework "Abstractions:" Different Ways to Describe the Same Thing (i.e. Enterprise)

The six interrogatives (Abstractions) are **primitive** and **comprehensive**. They are primitive in the sense that each interrogative is different from all the others, varies independently from all the others and all must be present to have a holistic description of the Enterprise. They are the minimum set. They are comprehensive in the sense that if you can answer all six of these questions about any subject (or object), from the answers to these six questions you can derive answers to any other question anyone asks about the subject (or object.) This is the total set. You need no other interrogative. In fact, there is no other primitive interrogative. This is a fixed list. It is not a hierarchy. The interrogatives are independent variables and the answers to these six primitive questions constitutes the total knowledgebase about the subject (or object) being described.

[1006] Webster's Dictionary defines abstract (v) as To remove, to separate, to consider apart from. It is in this sense that I chose to call the columns Abstractions.

Since the six Abstractions of the Framework are primitive and comprehensive (they are a fixed list, not a hierarchy) and the six Perspectives of the Framework are primitive and comprehensive (they are a fixed list, not a hierarchy), then each of the Cells of the Framework is primitive and the total set of Cells is comprehensive and it is "normalized" (one fact in one place).

Rules of the Framework

Rule 1: Do Not Add Rows or Columns to the Framework

Thousands of years of linguistic experience, irrespective of language or culture, would establish that Who, What, When, Where, Why and How are the six primitive interrogatives[1007]. If you can answer all of these six questions, from the answers to these questions, you can derive answers to any other question about the subject (or object) being described that anybody can construct. That is, the answers to these primitive questions would constitute the total knowledgebase for the subject (or object) you are describing.

These are the primitives, that is, you must have all of them to be complete. They are also comprehensive, that is, additional interrogatives add no new information. In fact, additional interrogatives introduce redundancies, which for classification purposes is anathema.

Both dimensions of the Framework classification scheme, the Perspectives and the Abstractions are both primitive and comprehensive. You need no additional Rows and no additional Columns to classify any primitive descriptive representation (knowledge) relevant for describing a complex object.

If I were to tell you that you should add another Column to the Framework, I would have to explain to you why I, John Zachman, have discovered that Who, What, When, Where, Why, and How were not adequate to elicit all the expressions of the complex object you need in order to describe it completely. Thousands of years of linguistic experience would suggest that that would be extremely presumptuous.

I don't have an explanation as clever or universally recognizable for the other dimension, the perspectives of the Framework. However, if I were to suggest adding another Row to the Framework, I would have to explain why, in the last 7,000 years, the people who build buildings haven't found the set of Artist's Rendering (sketches), Architect's Drawings, Architect's Plans, Contractor's Plans and Sub-Contractors' Plans to be adequate.

[1007] Ibid.

In short, adding Rows or Columns to the Framework would denormalize the classification scheme because, adding Rows or Columns would introduce redundancies or discontinuities.

The Framework, as it stands with no modifications, classifies all of the primitive descriptive representations (the total knowledgebase) relevant for describing an object, *any* object.

Rule 2: Each Column Has a Simple Generic Model.

Each Column of the Framework is descriptive of a single, independent variable within the analytical target, in our case, the Enterprise. Therefore, the basic generic model of any one Column is very simple: the variable (abstraction) it represents as related to itself.

Single Variable			Basic Generic Model
Column 1:	What	(Things)	Thing - Relationship - Thing
Column 2:	How	(Process)	Process - Input/Output - Process
Column 3:	Where	(Location)	Node - Line - Node
Column 4:	Who	(People)	People - Work - People
Column 5:	When	(Time)	Event - Cycle - Event[1008]
Column 6:	Why	(Motivation)	End - Means - End[1009]

For example, the generic model for all of the Cells in Column 1 is going to be Thing – Relationship – Thing. The generic model for all the Cells in Column 2 is going to be Process – Input/Output – Process. Etc., etc. Open Figure 13. Framework “Abstractions”: Generic Descriptions and Enterprise Equivalents.

[1008] Event equals a point in time, Cycle refers to a length of time.

[1009] End is the desired state, for example, objective. Means is a course of action employed to realize the end state, for example, strategy. At Row 3, the Ends/Means are embodied in the concept of a business rule where, according to the definitions of the Business Rules Group, the structural assertion is the expression of an End whereas action assertion is an expression of a Means. (See www.businessrulesgroup.org for Business Rules document.)

Generic Descriptions

"What" Material Description	"How" Functional Description	"Where" Spatial Description	"Who" Operational Description	"When" Timing Description	"Why" Motivation Description
Structure Generic Model *Thing - -Thing* *-Relationship-*	*Transform* Generic Model *Input - -Output* *-Process-*	*Flow* Generic Model *Site - -Site* *-Link-*	*Operations* Generic Model *People- -People* *-Work-*	*Dynamics* Generic Model *Event - -Event* *-Cycle-*	*Motivation* Generic Model *Ends - -Ends* *-Means-*

Enterprise Models

Data Model	Process Model	Network Model	Work Flow Model	Dynamics Model	Motivation Model
Entity - -Entity *-Relationship-*	*Input - -Output* *-Process-*	*Node - -Node* *-Line-*	*People- -People* *-Work-*	*Event - -Event* *-Cycle-*	*Obj. - -Obj.* *-Strategy-*

Figure 13. Framework "Abstractions:" Generic Descriptions and Enterprise Equivalents.

Rule 3: Each Cell Model Specializes Its Column's Generic Model.

The specific model for any given Cell will have to be customized to the constraints, the semantics, the vocabulary, the terms and facts of the Row's perspective. Furthermore, considering that the Cell description forms the baseline for managing change, the (meta) model will have to express all of the concepts affected by changes to that Cell model. Therefore, the specific (meta) model for a given Cell will start with the generic, columnar model, be adjusted it for the Row's semantic constraints and then it *might* have to be extended to accommodate all of the relevant concepts for expressing the constraints of the Cell's Row Perspective as well as for managing change to the Cell model, itself.

Rule 3, Corollary a: Level of Detail Is a Function of a Cell, Not a Column

Level of detail is a function of a Cell, not the Column, because from Cell to Cell a transformation is taking place through application of a different set of constraints, not simply addition of detail. Therefore, what is making a lower Row Cell different from a higher Row Cell in the same Column is NOT the level of detail. The Cells in different Rows of the same Column are different because they are actually models of different things.

That is, for any given Cell, you could have a high level of detail model, a medium level of detail model, or an excruciating level of detail model. Level of detail does not necessarily increase from Cell to Cell down a Column. Level of detail increases *within* a Cell. Practically speaking, the nature of the work you are trying to do at Row 5 necessarily would demand excruciatingly detailed models, and therefore, by definition, the Row 5 models are likely to be expressed at excruciatingly detailed levels.

A better way to envision level of detail is to separate the Rows so they don't appear contiguous. It is a transformation that is taking place from Row to Row. Excruciating level of detail of a higher Row is NOT a high level of detail of a lower Row. There should be a consistency between high level of detail slivers of all of the Rows. That is, a high level of detail of Row 1 should be consistent with a high level of detail of Row 2 which should be consistent with a high level of detail of Row 3 and of Row 4 and of Row 5. Open Figure 14. Level of Detail is a Function of a Cell, NOT a Column.

Another factor related to increasing levels of detail is that more detail is not necessarily hierarchical in nature. For example, clearly, more detail in the Cells of Column 1 is not hierarchical. It is a bill-of-materials. More detail in the Cells of Column 2 is hierarchical. None of the remaining Columns are hierarchical in nature.

Cells of:	**Level of detail:**
Column 1(Things)	Bill of Materials
Column 2 (Process)	Hierarchical
Column 3 (Location)	Geometric
Column 4 (People)	Matrix (Relational)
Column 5 (Time)	Cyclical
Column 6 (Motivation)	Complex Network

It is fortuitous that only the Process models are hierarchical because hierarchical decompositions result in unidentifiable redundancies in the detail which makes normalization difficult if not impossible. Also fortuitously, redundancy in the processes is only inefficient, not normally catastrophic. That is, doing the same Process different ways within an Enterprise is not efficient but normally, if the inputs and outputs are the same or at least acceptable at a appropriate level of decomposition, it won't cause the Enterprise to fail nor absorb inordinate amounts of energy or resources after the fact in compensation.

Rule 4: No Meta Concept Can Be Classified Into More than One Cell.

The Framework constitutes a clean classification system, that is, it is normalized. Each of the Columns is unique. Each of the Rows is unique. Therefore, each of the Cells is unique. No meta concept can be classified into more than one Cell. There is no redundancy. This is the one fundamental factor that makes the Framework a good analytical tool.

The logic of the Framework is never going to change. Although the instantiation of the models of the Framework (the CONTENTS of the models) may change as the technology changes, the business changes, the architectural state of the art changes, the tools change, the methodologies change, the Age changes, the people change, the culture changes, the regulations change and the political parties change ... the Framework is not going to change. ξξIt is primitive, comprehensive, natural and it is clean (normalized). In fact, it has been stable through the ages. In fact, if you go back and look at Aristotle's classification system, it has rather amazing similarities.

One more time: Do not add Rows or Columns to the Framework.

Rule 5: Do not Create Diagonal Relationships Between Cells.

First, the fact that the Owners, the Designers, the Builders and the Sub-Contractors are all using the same English word to mean entirely different Things, creates a very confusing communication problem. An elaborate example of this phenomenon is described below in the section entitled, Row to Row Transformations – An Example, but briefly, when the CEO uses the word "employee" what he or she has in mind is an employee, that is a living, breathing human being whereas, when a Programmer uses the word "employee" what he or she has in mind is a character string that has so many digits and such and such name designator like empno. The point is, empnos (a character string) and human beings (employees) are two different things.

The people in the various perspectives in the Enterprise may all be speaking English (or, whatever) and using the same words, and may be actually talking to each other, but they may not be close to communicating with each other. And, the greater the Row gap between the potential communicators, the greater the potential communications problems. For example, General Managers (Rows 1, 2) negotiating with Programmers (Row 5) over the design of the business, leaves a lot of room for mis-communication and its attendant risk. They may think they are communicating with each other because they are both speaking the same language and using the same words. However, the meaning of the words they are using may be so diverse that speaking the same language only creates the illusion of communication.

The semantic discord created by different people with different perspectives using the same word to express entirely different concepts elicits a practical analytical rule with regard to the Framework, never create a diagonal relationship between Cells.

Because of the semantic discontinuity, diagonal relationships leave gigantic holes for misinterpretation. You think you are communicating, but you probably are not.

Every Cell is related to every other Cell in its Row. Also, every Cell is related to the Cell above and the Cell below in its Column. Using only horizontal and vertical mappings avoids misinterpretation.

The structural reason for banning diagonals is because the cellular relationships are transitive. Changing a Cell logically may impact the Cell above and the Cell below in the same Column and any/every other Cell in the same Row. When things start changing, the only way to manage the impacts of the change is to manage the vertical and horizontal relationships. Because there is no structural logic defining diagonal relationships, the problem of change management approximates infinity. There are infinite diagonal combinations and permutations. Therefore, the way to address a changing Enterprise is to project the impacts of the change vertically and horizontally when any one Cell changes, and then, determine the changed diagonal relationship between that Cell and other Cells in different Rows and different Columns by inference.

Rule 6: Do Not Change the Names of the Rows or Columns

Do not change the names of the Rows or Columns, either in the generic Framework or in the Enterprise specific Framework.

Generic Framework

Rows: Scope, Owner, Designer, Builder, Out-of-context, Product
Columns: What, How, Where, Who, When, Why

Enterprise Specific Framework

Rows: Scope, Models of the Business, Systems Models, Technology Models, Detailed Representations, Functioning Enterprise
Columns: Data, Function, Network, People, Time, Motivation

Although I *may* entertain the necessity to change the name of a *Cell* because of common usage at some point in the future, I do not believe I would ever again consider changing the name of a Row or Column. Clearly, the logic of the Framework has never changed, and I am confident that it will never change. There is enough confusion out there in the primitive discipline of Enterprise Engineering and Manufacturing that I don't think it is advisable to have different-appearing, but same Frameworks with different Row and Column names floating around. In fact, I would argue that it is imperative to standardize our communications if we are ever going to advance the state of the art in Enterprise Architecture necessary to accommodate the Information Age Enterprise.

There is one more, maybe the most critical, reason not to change the name of the Rows and Columns. If you happen to change not only the name but the *meaning* of the Row or Column, now you have changed the basic logic structure of the Framework. It would no longer be the (quote) Zachman Framework. For example, if you change the meaning of one of the primitive interrogatives (What, How, Where, Who, When, Why), you no longer have the complete set nor do you have a normalized set. You have de-normalized the

Framework and at the same time made it less than complete. It would no longer be comprehensive.

Do not, I say again, DO NOT change the names of the Rows or Columns for the same reason you DO NOT want to add Rows or Columns. Changing the names may change the fundamental logic structure.

Rule 7: The Logic is Generic, Recursive.

The logic of the Framework is generic. As discussed above, the classification scheme of both axes was established quite independently of their application in the Framework. I learned about the Framework classification logic by empirically observing physical objects like airplanes, buildings, battleships, locomotives, computers, etc. Therefore, clearly, the Framework logic can be used to classify descriptive representations of physical objects, any physical objects.

Similarly, the Framework can be used to classify the descriptive representations of conceptual objects like Enterprises or Departments within Enterprises, or Projects within a Department or (computer) Programs within a Project, or multiple Enterprises. This characteristic of the Framework is discussed extensively in Chapter 13, Meta-Frameworks and Chapter 15, Multiple Frameworks in the Same Enterprise.

The Framework could be used to classify the descriptive representations of a Cell of the Framework. In this sense, it is like a fractal.

The Framework is generic. It can be used to classify the descriptive representations of anything and therefore to analyze anything relative to its architectural composition. It is recursive. It can be used to analyze the architectural composition of itself. The Framework is inert. It doesn't know what it is being used to analyze. Only the analyst knows the analytical target and establishes the boundaries of the analysis. The analytical boundaries selected for analysis have far-reaching implications and these issues are discussed in subsequent sections of this work.