Architecture is Architecture is Architecture PRINT

Architecture Is Architecture Is Architecture

By
John A. Zachman
March 2007 Reprint

There appears to be a gross misunderstanding about Architecture, particularly in the information technology community. Many people seem to think that an implementation, an end result, is Architecture. To use an Architecture and Construction example, many people think that the Roman Coliseum is Architecture.

This is NOT Architecture!

This is the RESULT of Architecture.

The Roman Coliseum is NOT Architecture. The Roman Coliseum is the RESULT of Architecture. The RESULT of Architecture is an instance of Architecture, an implementation. In the end result, the implementation, you can see an instantiation of the Architect's Architecture. If an Architect had not created the descriptive representations (the Architecture) of the Roman Coliseum, they could not have built the Roman Coliseum. They couldn't have even ordered the stones they required in order to build the Coliseum without the Coliseum Architecture which had to be created long before the Coliseum was constructed.

Architecture is the set of descriptive representations that are required in order to create an object. If you can't describe it, you can't create it. Also, if you ever want to change the object you created, Architecture *constitutes the baseline for changing* the object once it is created, that is, it is the baseline for changing the object IF you retain the descriptive representations used in its creation and IF you ensure that the descriptive representations are always maintained consistent with the instantiation.

If the object you are trying to create is so simple that you can see it in its entirety at a glance and remember how all of its components fit together at excruciating level of detail all at one time, you don't need Architecture. You can "wing it" and see if it works. It is only when the object you are trying to create is complex to the extent that you can't see and remember all the details of the implementation at once, and only when you want to accommodate on-going change to the instantiated object, that *Architecture is IMPERATIVE*. Once again, without Architecture, you are not going to be able to create an object of any complexity and you won't be able to change it (that is, change it in minimum time, with minimum disruption and minimum cost).

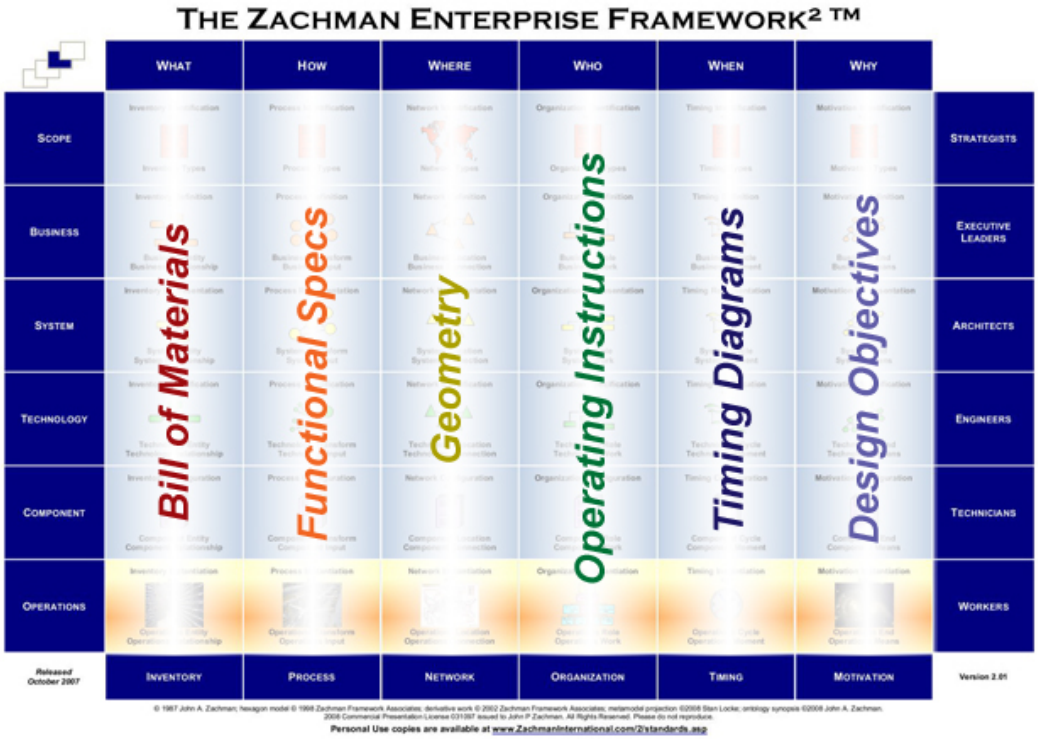
So, the question is, what constitutes the set of descriptive representations relevant for describing an object such that you can create it and change it with minimum time, disruption and cost?

The answer lies in several hundred years of empirical experience learning how to create and change complex industrial products.

There is a universal[1] set of descriptive representations for describing any or all industrial products. The sets of descriptions includes:

- Bills of Material (What)
- Functional Specs (How)
- Drawings (Where)
- Operating Instructions (Who)
- Timing Diagrams (When)
- Design Objectives (Why)

Abstractions



I have labeled these sets of descriptions "*Abstractions*" in the sense that out of the total set of relevant descriptive characteristics of the object, we "abstract" one of them at a time for producing a formal, explicit, description[2]. The Abstractions are universal in the sense that they are common to all industrial products as illustrated below:

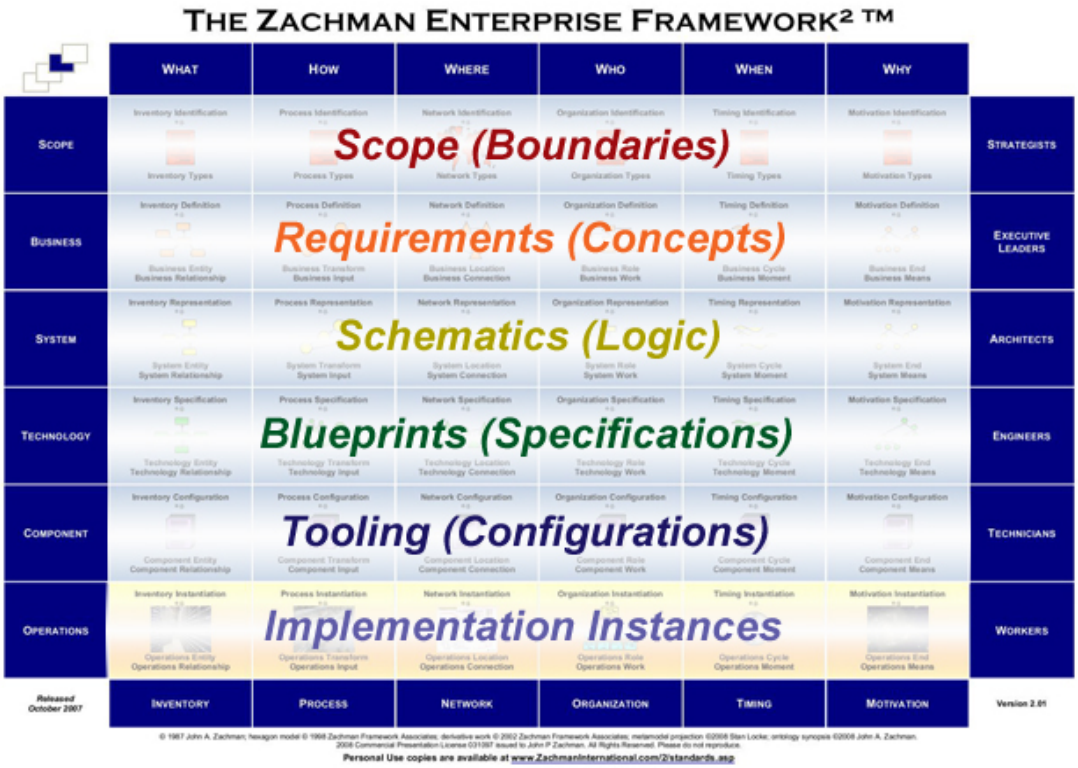
- e.g. the Material Abstraction (*WHAT* it's made of)
  - Airplanes have Bills of Material.
  - Locomotives have Bills of Material.
  - Computers have Bills of Material.
  - All Industrial Products have Bills of Material.

- e.g. the Functionality Abstraction (*HOW* it works)
  - Airplanes have Functional Specs.
  - Locomotives have Functional Specs.
  - Computers have Functional Specs.
  - All Industrial Products have Functional Specs.
- e.g. the Geometry Abstraction (*WHERE* the components are)
  - Airplanes have drawings.
  - And so on... and so on... and so on.

By the same token, all industrial products have:

- Scoping Boundaries (Strategists)
- Requirements (Concepts) (Owners)
- Schematics (Engineering descriptions) (Designers)
- Blueprints (Manufacturing Engineering descriptions) (Builders)
- Tooling Configurations (Implementers)
- Implementation Instances (Operators)

Perspectives



I have labeled this set of descriptions "*Perspectives*" in the sense that each of the Abstractions are created uniquely for different audiences[3]. Each of the Abstractions have five different, I say again, "different" manifestations, depending upon the Perspective of the intended audience for whom the Abstraction is created.

- e.g. Requirements (the Owner's Perspective)
  - Airplanes have Requirements.
  - Locomotives have Requirements.
  - Computers have Requirements.
  - All Industrial Products have Requirements.

- e.g. Schematics (the Designer's Perspective)
  - Airplanes have Schematics.
  - Locomotives have Schematics.
  - Computers have Schematics.
  - All Industrial Products have Schematics.

- e.g. Blueprints (the Builder's Perspective)
  - Airplanes have Blueprints.
  - And so on... and so on... and so on.

Why would anyone think that the descriptive representations for enterprises are going to be any different than the descriptive representations of anything else that has ever been created?

In fact, we, the ENTERPRISE Engineering and Manufacturing community (of which I/S is a part) have been reinventing the same descriptive representations that have already been invented by the older disciplines of Engineering/Manufacturing and Architecture/Construction, only we are putting our own names on them.

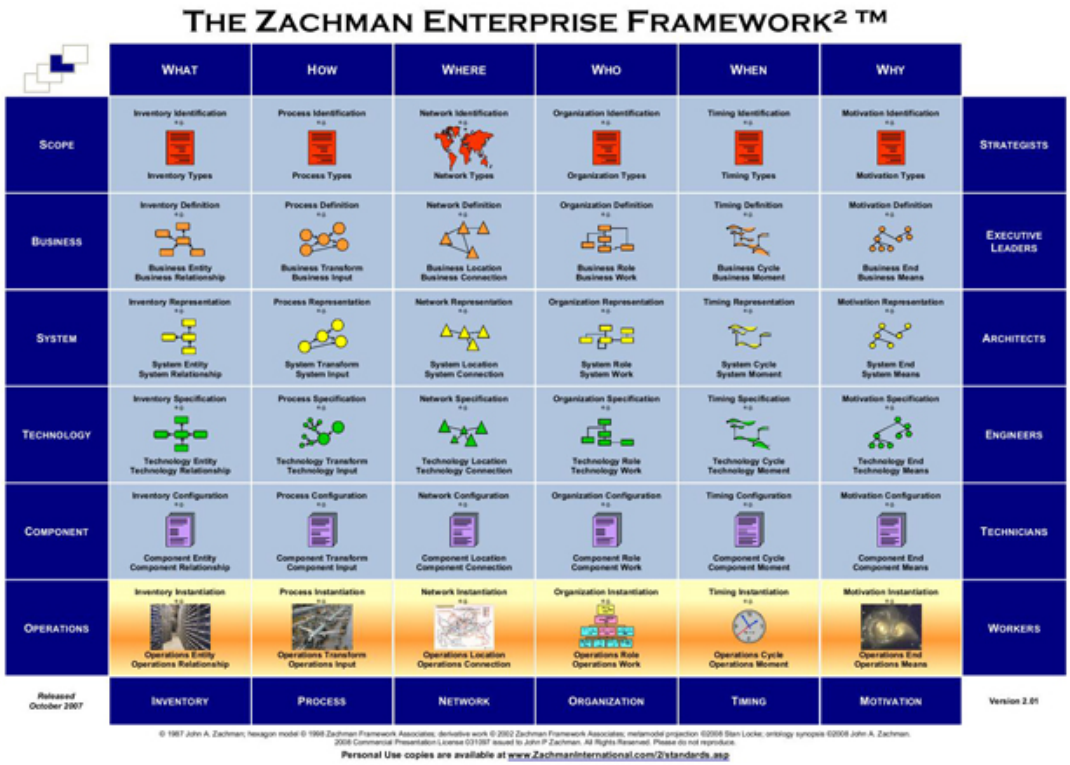
Here are the Enterprise equivalent descriptive representations:
e.g. Enterprise Descriptive Equivalents of Abstractions
Semantic Structures equal Bills of Material (Semantic Models ARE Bills of Material)
Process Models (or better, "Transformation" Models) equal Functional Specs
Distribution Models (Geography) equal Geometry (Drawings)
Work Flow models equal Operating Instructions
Dynamics Models or "Control Structures" (or better, "Timing" Models) equal Timing Diagrams
Design Objectives equal Design Objectives

By the same token:

e.g. Enterprise Descriptive Equivalents of Perspectives
Scoping Models equal Scope Boundaries (CONOPS or Concepts Packages)
Models of the Business (Concepts) equal Requirements
Models of the Systems (Logic) equal Schematics (Engineering Descriptions)
Technology Models (Technology Constrained Models) equal Blueprints (Manufacturing Engineering Descriptions)
Tooling Configurations equal Tooling Configurations
and
The Enterprise implementation equals the Industrial Engineering Product instantiation.

Therefore, **ENTERPRISE ARCHITECTURE**[4] **is the total set of intersections between the Abstractions and the Perspectives that constitute the total set of descriptive representations relevant for describing an Enterprise**:

ENTERPRISE ARCHITECTURE



This is the same total set of descriptive representations relevant for describing airplanes, locomotives, buildings, computers, all industrial products. I simply put the Enterprise names on the descriptive representations because I was interested in engineering and manufacturing Enterprises.

The Framework for Enterprise Architecture™ (***The Zachman Framework™***) is simply a schema, a classification scheme for descriptive representations of anything (I put Enterprise names on the descriptions and their contents – the metamodel) such that the schema is "normalized", that is, no one (meta) fact can show up in more than one Cell.

THE ENTERPRISE itself is the implementation, the instantiation, the End Result of doing Enterprise Architecture, assuming that any Enterprise Architecture has been done. I would observe that over the period of the Industrial Age until now, all airplanes, all locomotives, all buildings, all industrial products have been architected. However few (if any) Enterprises have been Architected. Up until now, Enterprises have simply happened... somehow. There may be many systems implementations, manual systems and/or automated systems, material handling systems (blue collar systems) and/or record keeping systems (white collar systems), a LOT of incoherence and discontinuity (ineffectiveness) and a LOT of compensation for that discontinuity (entropy, inefficiency). There is no Architecture. There are no **"Primitive" Models**[5]. There is no baseline for managing change. No Enterprise engineering has been done. Enterprise parts have been manufactured... but the Enterprise parts are not fitting together.

I predict that over the period of the Information Age, the next one or maybe two hundred years, all Enterprises will be Architected. The same factors that drove the formalization of Architecture for industrial products in the Industrial Age will drive the formalization of Architecture for Enterprises in the Information Age: **Complexity and Change**. We are already beginning to see the trend.

My observation is, **Architecture is Architecture is Architecture**. What Architecture is, is not arbitrary and it is not negotiable. Architecture is the total set of intersections between the *Abstractions* and the *Perspectives* that constitute the set of relevant descriptive representations for any object to be created.

If you cannot show me the Bill of Materials quite independent from Functional Specs quite independent from Geometry quite independent from Operating Instructions... etc., etc...

...And if you cannot show me Requirements quite independent from Schematics quite independent from Blueprints quite independent from Tooling Configurations... etc... etc...

...I do not believe you are doing Architecture work (Engineering). A single variable model, that is, one Abstraction by one Perspective, a "Primitive" model, is the **raw material for doing Engineering**. If you have no "Primitive" models, you have no raw material for doing Engineering and therefore, you are not doing Engineering (that is, you are not doing "Architecture").

In contrast, implementations, that is Manufacturing, the creation of the end results, are the instantiation of composite, multi-variable models, that is, models comprised of more than one Abstraction and/or more than one Perspective. A manufactured part (Material) has some Functionality in some Geometric location for some Operation at some Time for some Objective. An instantiation, by definition, is a "Composite[6]."

The question turns out to be, how did you create the composite, implementation instance? From Primitive models you have engineered from the perspective of the Enterprise? (Architected?) Or, did you simply create the Composite to produce an implementation ad hoc to whatever you were implementing (i.e. it was implemented, but NOT architected – i.e. NO Primitives)? And... is the Composite you created the whole complete object you are trying to

create (the whole airplane, or whole locomotive or whole Enterprise) or is the composite just a part of the whole thing (a wing or a boiler or a "system").

Once again, **if you cannot show me "Primitive" models, I know that you are not doing Architecture (Engineering). You are doing implementations (Manufacturing)**. And, if you are not creating "Enterprise-wide" Primitives, I know you are risking creating implementations that will not integrate into the Enterprise as a whole. You can manufacture parts of the whole iteratively and incrementally... however, they must be engineered to fit together or they are not likely to fit together (be integrated). You can even do the engineering, the Architecture, iteratively and incrementally, but in this case you must do something over and above building incremental, iterative primitives to mitigate the risk of misalignment and disintegration. *Enterprise-wide integration and alignment do not happen by accident. They must be engineered (Architected).*

If one thinks that an implementation, a result, a Composite model is Architecture, (whether it is the whole thing or only a part of the whole thing), then this is probably contributing to the misconception that, for example, the Roman Coliseum is Architecture.

The whole finished product, the end result, is the total agglomerate instantiated Composite of all the Abstractions and all the Perspectives. If one’s perception that the end result is Architecture, there is little wonder why Enterprise Architecture, that is, ENTERPRISE ARCHITECTURE (as in Enterprise-WIDE Architecture) is perceived to be big, monolithic, static, inflexible, unrealistic, impossible and generally unachievable therefore creating a DIS-incentive for even attempting Enterprise Architecture.

NO!!! IMPLEMENTATIONS ARE NOT ENTERPRISE ARCHITECTURE!!! Implementations are the result of architecture... If any architecture has ever been done!

If we ever want the Enterprise to be engineered so it is "lean and mean," so that it meets all the requirements of the "Owners," so that it is completely effective and efficient, so that it is integrated, so that it is dynamic, so that we can create new instances (implementations) on demand by assembling them to order from the Primitive constructs we already have in inventory, that is, so that we can "assemble the Enterprise to order" (in Manufacturing, "mass customization"), we have to start working on the *raw material* for doing Engineering, the single variable, "Primitive" models... **ARCHITECTURE, ENTERPRISE Architecture.**

The manufactured RESULT - (NOT the Architecture).

YES!!!... we will have to continue to satisfy current demand for implementations by building Composite implementation constructs in the short term. BUT, as we get some Primitives engineered (Architected) and into inventory, we can stipulate that any Composite models to be constructed MUST be constructed from the components of the architected Primitives we already have in inventory. In that fashion, over some period of time, we could migrate (maybe "evolve") out of the disintegrated, discontinuous, inflexible legacy environment into an Architected, coherent, flexible, dynamic, optimized *Enterprise*.

We likely could achieve the quality and longevity ascribed to Boeing 747's or hundred story buildings or other high quality, long-lasting, superior performing Industrial Age, complex engineering products that we have learned how to engineer over the last few hundred years.

Otherwise, nothing will have changed... we will just continue doing more of the same... building and running systems (legacy implementations, manual or automated, blue collar or white collar) and it doesn't make any difference what technologies we will be using. **It is not a technical issue.** It is an ENGINEERING issue, an *ENTERPRISE engineering* issue.

Are we going to start doing Enterprise engineering work (building Primitive models, i.e. Architecture)... or are we simply going to continue doing Enterprise manufacturing (building composite implementations, i.e. building and running systems... legacy)?

I would observe that it was Einstein that said something like, "keeping on doing the same thing and expecting different results is one definition of insanity."

Endnotes

1 The names of these descriptive representations may change slightly based on industry but the concepts represented are consistent. Furthermore, in some industries for some products, they may well be willing to assume the risks of not formalizing all of the relevant descriptive representations. Back to Article

2 For an exhaustive discussion of "Abstractions" see the Zachman eBook: "The Zachman Framework: A Primer for Enterprise Engineering and Manufacturing." Back to Article

3 For an exhaustive discussion of "Perspectives" see the Zachman eBook: "The Zachman Framework: A Primer for Enterprise Engineering and Manufacturing." Back to Article

4 For an exhaustive discussion of "Enterprise Architecture" see the Zachman eBook: "The Zachman Framework: A Primer for Enterprise Engineering and Manufacturing." Back to Article

5 For an exhaustive discussion of "Primitive Models" see the Zachman eBook: "The Zachman Framework: A Primer for Enterprise Engineering and Manufacturing." Back to Article

6 For an exhaustive discussion of "Composite Models" see the Zachman eBook: "The Zachman Framework: A Primer for Enterprise Engineering and Manufacturing." Back to Article

Back